# Engineering Self-Managed Adaptive Networks

*Daphne Selen Tuncer*

A dissertation submitted in partial fulfilment
of the requirements for the degree of
**Doctor of Philosophy**
of
**University College London**.

Department of Electronic & Electrical Engineering
University College London

2013

I, Daphne Selen Tuncer, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.



Department of Electronic & Electrical Engineering

University College London

# Abstract

In order to meet the requirements of emerging services, the future Internet will need to be flexible, reactive and adaptive with respect to arising network conditions. Network management functionality is essential in providing dynamic reactiveness and adaptability but current management approaches have limitations which prevent them from meeting these requirements. In search for a paradigm shift, recent research efforts have been focusing on autonomic/self-management principles, whereby network elements can adapt themselves to contextual changes without any external intervention through adaptive and flexible functionality.

This thesis investigates how autonomic principles can be extended and applied to fixed networks for quality of service and performance management. It presents a novel resource management framework which enables intelligence to be introduced within the network in order to support self-management functionality in a coordinated and controllable manner. The proposed framework relies on a distributed infrastructure, called the management substrate, which is a logical structure formed by the ingress nodes of the network. The role of the substrate is illustrated on realistic resource management application scenarios for the emerging self-managed Internet. These cover solutions for dynamic traffic engineering (load balancing across multiple paths), energy efficiency and cache management in Internet Service Providers.

The thesis addresses important research challenges associated with the proposed framework, such as the design of specific organisational, communication and coordination models required to support the different management control loops. Furthermore, it develops, for each application scenario, specific mechanisms to realise the relevant resource management functionality. It also considers issues related to the coexistence of multiple control loops and investigates an approach by which their interactions can be managed. In order to demonstrate the benefits of the proposed resource management solution, an extensive performance evaluation of the different mechanisms described in this thesis have been performed based on realistic traffic traces and network topologies.

# Acknowledgements

First of all, I would like to express my gratitude to my supervisor, Professor George Pavlou, for giving me the opportunity to pursue a Ph.D. I thank him for his help and guidance, and his confidence in the work that I have been carrying out in the context of this Ph.D. I would also like to thank a lot Dr. Marinos Charalambides for all his help and support, and numerous discussions about the research presented in this thesis. This was a pleasure to work together.

I also thank Dr. Ning Wang, for all his insightful comments and discussions about traffic engineering and multi-topology routing, and Dr. Raul Landa, for all his suggestions about different research questions tackled in this thesis. I would like to express my appreciation to Dr. Stuart Clayman for all his help with programming and with the different software tools, and for the discussions about the Ph.D. in general. This was very supportive. Thanks to Dr. Ioannis Psaras for providing insightful feedback regarding this thesis. I also thank a lot our previous master students, Mr. Prashant Jain and Mr. Hisham El-Ezaby, for their help with the experiments. Many thanks as well to all my colleagues of the NSRL group.

I would also like to thank the external and internal examiners of my Ph.D., Professor Joseph Sventek and Dr Yiannis Andreopoulos, for all their valuable comments and feedback concerning this Ph.D.

Finally, I wish to thank my family and my friends for their permanent encouragement and support throughout my Ph.D. life.

The work carried out in this thesis was partly funded by:

- The EU FP7 COMET project (No. 248784).
- The EU FP7 Flamingo Network of Excellence project (No. 318488).

# Contents

# List of Figures

# List of Tables

# Abbreviations

**BL** Bundled Link

**CDN** Content Delivery Network

**CL** Control Loop

**DE** Deciding Entity

**DIT** Demand Information Table

**ECMP** Equal Cost Multi-Path

**EM** Energy Management

**EM-CL** Energy Management Control Loop

**GCP** Global Content Popularity

**GDI** Geographic Distribution of Interests

**IE** Intermediate Entity

**IP** Internet Protocol

**ISP** Internet Service Provider

**LB** Load Balancing

**LB-CL** Load Balancing Control Loop

**LC** Line Card

**LIT** Link Information Table

**MAPE-K** Monitor Analyse Plan Execute-Knowledge

**Max-u** Maximum utilisation

**MPLS** Multi-Protocol Label Switching

**MS** Management Substrate

**MT-ID** Multi-Topology Identifier

**MTR** Multi-Topology Routing

**OSPF** Open Shortest Path First

**PoP** Point of Presence

**RTT** Round-Trip Time

**TCP** Transmission Control Protocol

**TSP** Travelling Salesman Problem

# Publications

- D.Tuncer, M.Charalambides, R.Landa, G.Pavlou, **More Control Over Network Resources: An ISP Caching Perspective**, in the Proceedings of the 9th IEEE/IFIP International Conference on Network and Service Management (CNSM'13), October 2013.
- M. Charalambides, D. Tuncer, L. Mamatas, G. Pavlou, **Energy-Aware Adaptive Network Resource Management**, in the Proceedings of IFIP/IEEE Integrated Management Symposium (IM'13), May 2013.
- D.Tuncer, M.Charalambides, G.Pavlou, N.Wang, **DACoRM: A Coordinated, Decentralized and Adaptive Network Resource Management Scheme**, in the Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS'12), April 2012.
- M. Charalambides, G.Pavlou, P. Flegkas, N. Wang, D. Tuncer, **Managing the Future Internet through Intelligent In-Network Substrates**, IEEE Network Magazine, Special Issue: Managing an Autonomic Future Internet, Vol. 25(6), November 2011.
- D.Tuncer, M.Charalambides, G.Pavlou, N.Wang, **Towards Decentralized and Adaptive Network Resource Management**, in the Proceedings of the 7th IEEE/IFIP International Conference on Network and Service Management (CNSM'11), mini-conference, October 2011.
- D.Tuncer, M.Charalambides, G.Pavlou, **An On-line Approach for Adaptive Resource Management in Future IP Networks**, in the Proceedings of the London Communication Symposium (LCS'11), September 2011.
- D. Tuncer, M. Charalambides, and G. Pavlou, **Towards dynamic and adaptive resource management for emerging networks**, in the Proceedings of 4th International Conference on Autonomous Infrastructure, Management and Security (AIMS'10), PhD Workshop, Burkhard Stiller and Filip De Turck (Eds.), Springer-Verlag, Berlin, Heidelberg, 93-97.

## Chapter 1

# Introduction

## 1.1 Context and Motivation

With the evolution of communication technologies and the emergence of new services and applications, developing approaches for the management of network resources with minimum human intervention has become a key challenge.

Today, networks are planned and configured in a predictive manner through offline traffic engineering [2] where the expected demand is calculated from previous usage and a specific routing configuration is produced, aiming to balance the traffic and optimise the use of network resources for the next provisioning period. Reactive approaches are not possible given the current nature of management systems that are external to the network and the resulting latency in learning about arising conditions and applied changes. As such, this external offline configuration approach can be highly sub-optimal in the face of changing or unpredicted traffic demand. The only current dynamic intervention relates to control functionality within network devices in order to deal with faults. For example, fast reroute control functions [3] direct traffic away from a failed component upon detecting a failure. But this is done in a predefined manner that does not take into account any other aspects related to the current state of the network and can have an unforeseen negative impact elsewhere.

This lack of adaptability to the network state makes the current Internet unable to cope with the requirements of emerging services and time-critical applications, characterised by highly demanding traffic, varying traffic patterns and quality of service requirements. To meet these requirements, the future Internet is expected to become more flexible, adaptive and reactive to traffic and network dynamics. Due to the static nature of current management approaches towards these emerging demands, a paradigm shift is required and new management approaches should enable the Internet to continuously optimise the use of its resources and to recover from problems, faults and attacks, without impacting the supported services and applications.

## 1.2 Problem Statement

To overcome the drawbacks and limitations of current approaches, research efforts have been applying the *autonomic computing* principles developed by IBM [4] to network management systems. *Autonomic computing* aims at reducing the need for human intervention in the management process through the use of control loops that continuously reconfigure a system according to high level objectives from an administrator. Autonomic self-managed networks should be able to support self-aware, self-adaptive and self-optimising functionality that will be able to adapt and react gracefully to changes through feedback control solutions. While functionalities with various degrees of automation are already implemented in today's networks and systems, these mainly apply to specific problem domains. To support the requirements of the future Internet, ongoing research efforts have been focusing on extending these functionalities to the various aspects of network management.

The objective of the work in this Ph.D. was to address, through realistic and specific in-network self-management applications, some of the main engineering challenges raised by the design and the implementation of self-managed adaptive networks. To achieve the objective, this work investigated how autonomic principles can be extended and applied to fixed backbone networks such that continuous resource optimisation can be supported through dynamic and adaptive network resource management functions.

## 1.3 Contributions

The design of an approach for adaptive resource management raises several research challenges. Unlike current approaches, a self-managed network should support decentralised decision-making and control functions. The decision-making process is distributed among sets of nodes in the network, each being responsible for the reconfiguration actions to take based on local feedback regarding the state of the network. The main benefits of this approach are robustness to failures and fast adaptation to changing conditions. However, as each node has only partial knowledge of the global information, inconsistencies between several independent decisions can occur, which may jeopardise the stability and the convergence of the global behaviour of the system. As such, management decisions need to be made in a collaborative manner and be harmonised toward a common objective.

Based on representative network resource management application scenarios, the work in this Ph.D. developed self-managed network resource optimisation functionality that addresses the issues of adaptive resource management. The contributions of this work are organised into five main chapters as follows.

**Management Substrate Framework:** In order to support adaptive resource management functionality, a distributed in-network resource management infrastructure was developed. The resource management decision process is distributed across the set of network edge nodes which are embedded with dedicated management logic that enables them to perform reconfigurations based on feedback regarding the state of the network [5][6]. The set of network edge nodes is organised into a management substrate structure that is used to exchange information for making and enforcing management decisions. In order to achieve consistency in the configuration decisions, these are made in a collaborative manner between nodes in the substrate. The use of different topology models to organise the nodes in the substrate was investigated [7]. Algorithms that take into account important criteria such as minimising the latency and the communication overhead among the substrate nodes were developed to compute the proposed structures. A quantitative and qualitative evaluation of the different structures in terms of cost and complexity was performed based on real network topologies. In order to validate the use of the proposed management infrastructure, this was applied to the following adaptive resource management application scenarios.

**Load Balancing-aware Adaptive Resource Management Application:** Current practices for managing resources in fixed networks mainly rely on offline approaches, which can be sub-optimal in the face of changing or unpredicted traffic demand. To cope with the limitations of these offline configurations, new traffic engineering schemes that can adapt to network and traffic dynamics are required. In this Ph.D., an intra-domain decentralised and dynamic traffic engineering approach for IP networks was developed. The proposed solution uses multi-topology routing as the underlying routing protocol to provide path diversity and supports adaptive resource management operations that dynamically adjust the volume of traffic routed across each topology [6][7]. Reconfiguration actions are performed directly by the network edge nodes themselves without relying on a centralised management system. In order to guarantee globally consistent operation, the network edge nodes coordinate through the management substrate to decide on the best sequence of actions to apply. Specific coordination mechanisms were developed according to the structure used to connect the nodes in the substrate. The performance of the proposed approach was evaluated using realistic network topologies and traffic traces, and the results showed that near-optimal network performance can be achieved in terms of resource utilisation in a responsive and scalable manner.

**Energy-aware Adaptive Resource Management Application:** Resource over-provisioning is common practice in network infrastructures. Coupled with energy unaware networking protocols, this can lead to periods of resource under-utilisation and constant energy consumption

irrespective of the traffic load conditions. Research in power saving techniques has recently received significant attention. Unlike the majority of previous research in the area, which has focused on centralised offline approaches, an online approach by which the capacity of the network can be adapted in a decentralised fashion was developed. Based on the modular architectures of modern IP routers, adaptation is achieved by configuring individual line cards to enter sleep mode [8]. In order to support the proposed scheme, the mechanisms developed for the load balancing resource management application were extended. In a similar fashion to the previous application, reconfiguration is performed periodically by intelligent network edge nodes that coordinate their actions through the substrate infrastructure in order to control the traffic distribution in the network, according to the current demand. The proposed approach was evaluated using real network topologies and traffic traces. The results showed that a significant gain can be achieved with the proposed scheme in terms of energy required to power the line cards, while preserving the connectivity of the network topology.

**Content Placement Management Application:** Management operations performed by Content Delivery Network providers consist mainly in controlling the placement of content at different storage locations and deciding the locations from which to serve client requests. Configuration decisions are usually taken by using only limited information about the carrier networks, and this can adversely affect network usage. This Ph.D. investigated an approach by which Internet Service Providers (ISPs) can have more control over their resources. This involves the deployment of caching points within their network, which can allow them to implement their own content placement strategies [9]. Lightweight strategies that can be used by the ISPs to manage the placement of contents in the various network caching locations according to user demand characteristics were developed. The proposed content placement management approach can be realised through the distributed in-network resource management infrastructure. Distributed cache managers coordinate between themselves to decide how to efficiently use the available caching space with the objective of better utilising the network resources. In order to evaluate and compare the performance of the proposed strategies in terms of network resource utilisation, different metrics were designed to model the content request distribution. The results demonstrated that the proposed metrics can provide useful indications regarding the performance one strategy can achieve over another and, as such, can be used by the ISP to improve the utilisation of network resources.

**Control Loop Interactions Management:** Specific control loop functionality was developed for each of the adaptive resource management application scenarios described above. However, a self-management solution with rich functionality would require multiple control loops

operating at the same time, which can interact with each other. These interactions can cause configuration instabilities and subsequently network performance degradation, especially in the presence of contradicting control loop objectives. In this Ph.D., the problem of interactions was investigated in the context of the two control loops that perform dynamic network resource reconfiguration for load balancing and energy management purposes. This research identified how potential configuration inconsistencies can arise and proposed an approach which can resolve these inconsistencies by determining configurations that satisfy both optimisation objectives in a sequential manner. The approach was evaluated using real network topologies and traffic traces, and the results were compared against those obtained by each control loop individually. It was demonstrated that, for the scenario considered, a performance trade-off can be achieved between the two optimisation objectives.

To summarise, this thesis proposes a new in-network management framework for adaptive and decentralised resource management and exemplifies its use based on three challenging application scenarios for the emerging self-managed Internet. It presents new approaches for online traffic engineering, energy efficiency management and cache configuration management, and shows how these can be used by Internet Service Providers to improve the utilisation of their network resources. Finally, it discusses important research issues raised by the coexistence of multiple resource management control loops and presents an approach by which their interactions can be managed.

## 1.4 Thesis Outline

The rest of this thesis is organised as follows. Chapter 2 provides the state-of-the art in the area of autonomic computing and network self-management. Chapter 3 introduces the distributed in-network resource management infrastructure to support adaptive network resource management functionality. Chapter 4 presents a novel traffic engineering approach that allows to control the traffic distribution in the network in an adaptive and decentralised fashion. In chapter 5, the proposed adaptive resource management scheme is extended to achieve energy savings. Chapter 6 presents a cache management approach with which Internet Service Providers can have more control over their network resources. Chapter 7 investigates the problem of interactions between the adaptive resource management applications for load balancing and energy management purposes. Finally, chapter 8 concludes this thesis and discusses future work.

## Chapter 2

# Related Work

## 2.1 Introduction

The evolution of information and communication technology (ICT) over the past thirty years has heavily influenced the life of the modern consumer. ICT plays today an important role in the way people interact that goes beyond simple technological solutions. This technological revolution has catered for a persistent demand in terms of new services and applications which are characterised by strict requirements in terms of availability, service quality, dependability, resilience and protection. This has resulted in increasingly complex networks and software systems that need to support heterogeneous applications, technologies and multi-vendor equipment. As such, the management of network infrastructures has become a key challenge both for the industry and for the research community. Despite the evolution of management technologies, most of the existing management applications have been applied to the management of individual devices [10]. New approaches to address the management of complex networks are therefore required.

Over the past ten years, the concept of *autonomic* systems has received significant attention from both the research community (e.g. [11][12]) and industry (e.g. [13][14]), who envisioned it as a key area of research for the design and development of management systems for the future Internet. The main objective of an *autonomic* system is to reduce human involvement in the management process through the use of feedback control loop solutions that continuously reconfigure the system in order to keep it in a desirable state. The *autonomic* or self-management paradigm has been considered as a way to manage complexity [11] according to the promise of "systems that can anticipate, diagnose and circumvent any impairments in their operation in an independent and autonomic fashion with little human intervention" [14]. Research efforts have focused on enabling self-management capabilities, whereby network elements can adapt themselves to contextual changes without any external intervention through adaptive and flexible

functionality.

In this chapter, a review of the related work in the literature is presented. In order to highlight the limitations of current network management approaches, a brief overview of network management technologies and frameworks is provided in section 2.2. Section 2.3 describes the principles of autonomic computing defined by IBM [13] and considered as a reference model to represent *autonomic* systems. Section 2.4 discusses how research efforts in the area of network self-management have used and extended these principles to overcome the limitations of current approaches. It presents the main engineering challenges and techniques that have been considered to enable network self-management. Finally, section 2.5 reviews current efforts towards integrating self-management functionality for adaptive resource management.

## 2.2 Network Management Overview

### 2.2.1 Principles

Network management can be described as the set of activities, methods, procedures, and tools that pertain to the operation, administration (i.e. resource allocation and usage), maintenance (i.e. repairs and upgrade), and provisioning of networked systems [15]. The different network management functions are traditionally divided into five main categories referred to as the FCAPS, i.e. Fault, Configuration, Accounting, Performance and Security management. Each category consists of various techniques and methods to control the operation of network elements so that desired objectives, for instance in terms of resource utilisation or quality of service, can be satisfied.

Today, most network management architectures follow similar principles [16]. Management operations are usually performed through a network management system (NMS) that interacts with the network devices (i.e. the managed system) according to network management protocols such as the Simple Network Management Protocol (SNMP) [17]. The NMS consists of a set of management entities (software and hardware elements) that implement and execute the management functionalities. More specifically, managed devices are equipped with software modules that control the device's environment. These enable, in particular, the managed devices to send notifications to the management entities under certain conditions, for instance in case thresholds are exceeded. Upon receiving notifications, the management entities execute a decision-making process to determine which actions to take. Examples of actions include operator notification, event logging, system shutdown, or automatic attempts at system repair [16]. Management entities can also periodically collect information from the network elements (e.g. device counters or performance parameters) in order to monitor certain variables and sub-

sequently determine whether new configurations are required. In such a case, the decisions of the management entities are translated into control actions that are directly applied to the managed devices.

### 2.2.2 Framework and Technologies

A wide range of technologies and approaches for network management have been developed over time. One of the key technologies is SNMP [17], which is a widely used protocol. Although it provides a range of functionalities, it has mainly been applied to some of the management areas only [10]. Configuration management, for example, is an area in which the use of SNMP has been limited. Other important tools are the management data models (e.g. the management information base (MIB)) that define and standardise specific technologies and protocols. In addition to SNMP, other key technologies have been considered in the network management community such as the OSI-SM which is mainly used in telecommunication environments [18], CORBA which is one of the most representative distributed object technologies, as well as XML and web-based approaches.

One of the main characteristics in the evolution of management approaches is the automation of the tasks involved in the management process. In the early ages of network management, the management process mainly relied on the expertise of the network administrator who was responsible for manually querying and configuring individual network devices. As network technologies evolved, however, such a manual process became highly inefficient. In order to alleviate the role of the human operator, and as such, improve the performance and reduce operational costs, the automation of management tasks has been progressively introduced in the management process.

The use of policies, in particular, has enabled the development of more efficient approaches for the management of large distributed systems (e.g. [19][20][21]). In policy-based management frameworks, different mechanisms are implemented to guide the operation of the network according to a set of rules that translate high level directives into low level policies that can control the configuration of the devices' operational parameters. Low level policies can for instance follow an *event-condition-action* policy model by which when *an event occurs and some conditions hold, then relevant actions are executed*. A well-known example of such a model is the Policy Language for Distributed Systems Management (PONDER) [22]. Policy-based management frameworks provide flexibility in managing network resources at run time [23]. A key issue in these frameworks, however, is to handle the conflicts that may arise between different policies, as these can jeopardise the stability of the managed system. In order to deal with this issue, conflict detection and resolution mechanisms have been proposed in [24][25].

Conflicts can be resolved, for example, by giving different priorities to the policies.

Despite the evolution of the management technologies and frameworks, especially through the partial automation of management tasks, today's network management approaches have limitations. Current approaches usually rely on offline and static management functionality executed through a network management system that resides outside the network. Due to their rigid nature, these approaches lack the ability for situation and context adaptability, and as such, are inefficient in dealing with emerging traffic conditions and failures. In order to support new services and applications, flexible, adaptive and reactive functionalities are needed.

## 2.3 Autonomic Computing

### 2.3.1 Self-Management Properties

Autonomic management principles were conceptually defined and formalised by IBM in its vision of *autonomic computing* [26]. These were further extended in [4] and [27] in which the main engineering challenges involved in the design of *autonomic computing* systems are discussed.

The essence of *autonomic computing* is inspired from the autonomous nervous system of the human body, which maintains an equilibrium between various biological processes (e.g. respiration, heart rate etc.) without any conscious effort [26]. More specifically, *autonomic computing* can be defined as computing systems that are able to manage themselves given high level objectives provided by the system administrators and that satisfy the four following properties referred as the "self" properties [4][27]:

- **Self-configuration**: The ability of the system to (re-)configure itself given high level specifications, so that the system can self-organise into a desirable structure and pattern.
- **Self-optimisation**: The ability of the system to automatically adapt to a changing environment in order to optimise its performance and efficiency (e.g. its cost).
- **Self-healing**: The ability of the system to automatically detect, diagnose and recover from faults, resulting either from internal or external errors.
- **Self-protection**: The ability of the system to protect itself against potential attacks or inadvertent failures.

Although these properties are usually considered separately, they can also be merged into a more general notion of self-maintenance that can be used to characterise *autonomic computing* systems [4].

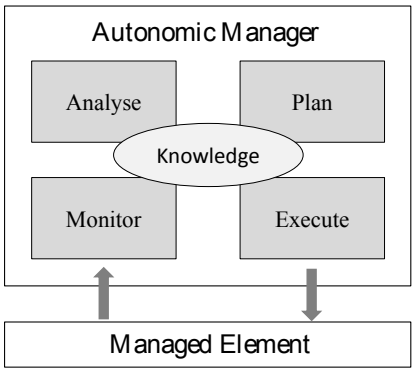



Figure 2.1: The MAPE-K control loop.

A parallel between the "self" properties and the properties of software agents was presented in [28]. According to the authors, basic agent properties such as autonomy, social ability (i.e. interaction between agents), reactivity and proactiveness [29] can describe some of the features of *autonomic computing* systems. Unlike autonomic systems, however, the "self-management" behaviour of software agents essentially relies on the interactions between the agents rather than on the agent itself, as highlighted in [30].

### 2.3.2 The MAPE-K Control Loop

The *autonomic computing* paradigm is realised through the introduction of autonomic managers that execute an autonomic management control loop to monitor and control managed elements and their operational environment [4]. The managed elements can represent any hardware or software resource, such as a storage disk, a Central Processing Unit (CPU), a database, a router queue or a fault localisation service [31]. Based on the monitored information, the autonomic managers can control the current state of the managed elements and decide whether a reconfiguration is required. More specifically, the autonomic management control loop consists of four components (the **M**onitor, **A**nalyse, **P**lan and **E**xecute components) and a common **K**nowledge repository that maintains information about the computing environment, the different management policies and the user context, as depicted in figure (2.1). This model, referred to as the MAPE-K control loop, is considered as a reference model in the context of autonomic management architecture [28].

The *monitor* component is responsible for collecting information from the operational environment and the managed elements. The collected information is then analysed by the *analyse* component that determines the current state of the managed element. The output of the analysis is provided to the *plan* component that decides whether any reconfiguration is required. In such a case, it determines the set of actions to perform, which are then translated into a set of configurations and enforced by the *execute* component.

### 2.3.3 The Autonomic Manager

From an architectural perspective, autonomic computing systems are represented as an interactive collection of autonomic elements, each performing a specific operational function [32]. Each autonomic element consists of (one or more) managed elements that execute the operational function and an autonomic manager that controls the configuration, inputs and outputs of the managed elements according to the MAPE-K control loop model. The autonomic manager is also used as the interface to interact with the other autonomic elements and with human operators. Some generic implementations of an autonomic element have recently been proposed in [33][34].

The autonomic manager is a software component that enables the translation and execution of high level objectives into low level actions (e.g. parameter updates). There has been a large body of work focusing on the use of policies in the context of autonomic management, e.g. [35][36][37][38]. Previous research has especially considered different policy models to capture and translate high level objectives into quantitative parameters, such as goal policies (e.g. in [39]) or utility function policies (e.g. in [40][41][42]). Utility function policies can be used to specify preferences. In that case, the different system states are quantitatively described by a utility function which maps each possible state to a real scalar value. The state that maximises the utility is then selected [40]. Utility function policies have been used in [41] in the context of autonomic resource allocation, and in [42] where an approach to orchestrate the interactions between autonomic elements is investigated.

## 2.4 Network Self-Management

### 2.4.1 From Autonomic Computing to Autonomic Networking

To overcome the limitations of existing network management approaches, research efforts have been extending the autonomic computing principles to network management systems, i.e. to the collective self-management of individual networking devices [12][43][44]. These have investigated how to apply self-management concepts to the various issues in the area of network management, such as fault, security, configuration, performance etc. [12]. According to the autonomic principles, network management systems are enhanced with self-aware, self-adaptive and self-optimising functionality, which is embedded within the network devices themselves. This enables the system to analyse and understand its environment, to decide upon the appropriate configuration actions, and to learn from previous decisions based on closed loop feedback control solutions. In order to perform the closed loop control functionality, sophisticated monitoring mechanisms are in charge of gathering network information, which can be used by

decision-making processes distributed across network nodes.

While significant research effort has been invested in self-management approaches over the last past ten years, the integration of autonomic features within the network environment has been a key research challenge for a long time. Some automated techniques for the configuration and management of networks are already in use today [45], such as the Transmission Control Protocol (TCP) control loop to adjust the congestion window, or failure detection and adaptation techniques in link-state routing protocols (e.g. Open Shortest Path First (OSPF)) to reroute packets towards new valid paths. These have, however, mainly focused on addressing specific issues in isolation. In contrast, current efforts in self-management go beyond these approaches in that they have been investigating how to integrate automated techniques into the different tasks of the management process, and as such, issues related to the interaction and coordination between independent control loops have become key research challenges.

As highlighted by several authors ([11][12][44][46]), research in the area of self-management have been characterised by the choice of the engineering challenges to target. Furthermore, it is interesting to note that no consensus has been reached in providing a formal definition of a self-management approach [14]. It is often agreed that a network management system is said to be autonomic if the management operations can be performed in such a way that the four "self" properties, presented in section 2.3, are satisfied [46][28]. In that case, the functionality traditionally found in higher levels of networked system management is relocated towards the network elements that can execute the management operations themselves.

According to the authors of the report [14], reference models are required in order to characterise any autonomic network architecture. These models would, in particular, identify how the communication between control loops is organised, the semantics of the information (i.e. knowledge) and how the management responsibility is distributed across the different entities. Recently, some qualitative and quantitative evaluation factors to characterise and compare self-management systems have been proposed in [46][31]. The main characteristics considered are the degree of reactivity/pro-activity of a system, the degree of adaptability, the degree of intelligence (i.e. ability to learn), the degree of awareness (i.e. knowledge about the environment and/or user context), as well as the degree of autonomy. In addition, it has been recently argued that an important evaluation factor to consider is the cost of an infrastructure [14]. Cost functions to evaluate the cost incurred by any self-management approach, especially in terms of computation and communication overhead, have been defined in [31].

### 2.4.2 Network Self-Management Architectures

Several approaches related to self-management have been proposed in the literature, e.g. [39][43][47][48]. Most of this work has focused on high level architectures and design of infrastructures only partially considering key research issues.

The most comprehensive self-management reference architecture for networks is FOCALE (Foundation Observation Comparison Action Learn rEason) [43][49]. The FOCALE framework integrates distributed self-management algorithms with policy-based management. It considers the use of information and ontological modeling to gather knowledge about network capabilities and constraints and to build an abstract representation of vendor specific functionalities. The autonomic features are incorporated into autonomic management elements that implement two control loops for maintenance and adaptation to changes in the environment, in business rules and/or in user requirements. In addition to FOCALE, other self-management architectures have been proposed in [39][47][50][51]. In [47], the authors focused on the design of a generic autonomic architecture for service delivery over IP networks. The proposed architecture was applied to self-management in the context of DiffServ/MPLS (Differentiated Services/Multi Protocol Label Switching) networks to which the concept of autonomic element was adapted. A key characteristic of this approach is to define everything in the network environment - from a TCP connection to an application - as a service and to consider the combination of service elements in the realisation of management tasks. The autonomic network management framework proposed in [39] relies on the use of different policy models to guide the management decision-making process. In order to describe how autonomic elements can react to changes in the environment, the introduction of abstract behavioural policies has been particularly considered. The role of the framework was illustrated in the context of the management of Quality of Service and Quality of Experience in multiservice IP networks. A resource management framework to support self-configuration was also investigated in [51], where two control loops, defined according to the MAPE-K model and that operate at different timescales, are implemented to perform the self-management functionality. Models of communication and interaction between the two control loops were analysed and applied to the design a new routing algorithm based on the concept of betweenness centrality. In [50], a management framework for autonomic applications in Grid environments was proposed. In this work, a specific semantic is used to describe the different applications which are represented as autonomic elements. The dynamic composition of autonomic elements is performed through a set of rules and mechanisms that rely on an agent infrastructure. Examples of the definition of rules are provided in the context of a forest fire management application.

Research initiatives for the implementation of self-management functionalities have also been the target of some European Union funded projects such as the Autonomic Network Architecture project (ANA) [52], the Autonomic Internet (AutoI) project [53] and the UniverSELF project [48]. Research efforts in the ANA project focused on the design of a meta architecture to support the co-existence and interworking of a large sets of network architectures ranging from personal area networks to large scale networks [52]. The proposed framework aims at providing functional scaling, i.e. to facilitate the integration of new network functionality both within a single domain and across domains through the development of appropriate levels of abstraction. Abstraction models are also considered in the AutoI project. In this project, the autonomic management architectural model consists of an abstraction of the management process according to five main functions, i.e. Virtualisation, Management, Knowledge, Service Enablers and Orchestration Planes, that are represented through distributed management systems running within the network. The distributed systems are envisioned to form a software-driven control network infrastructure that will operate on top of all current network and service infrastructures. In the UniverSELF project, the proposed management architecture is called the Universal Management Framework. It is defined in the form of functional blocks and interfaces that ensure the trustworthy integration, operation and interworking (e.g. conflict avoidance and knowledge sharing) of multiple autonomic control loops, encapsulated in management abstractions that represent the autonomic elements and that are called Network Empowerment Mechanisms [54].

### 2.4.3 Techniques and Methods for Autonomic Features

In order to realise autonomic network management systems, research efforts in the literature have considered principles and techniques from various disciplines, such as agent-based approaches, control-theory tools or bio-inspired solutions.

**The Knowledge Plane** A new paradigm for networks in the form of a *Knowledge Plane* was presented in [55]. The *Knowledge Plane* is defined as an extra plane between the data plane and the management plane and is described as a "pervasive system within the network that builds and maintains high level models of what the network is supposed to do in order to provide services and advice to other elements of the network". In that work, the network is envisioned as an intelligent cognitive system that can configure and operate itself given high level instructions. The *Knowledge Plane* can be thought as a separate cognitive framework, or as architectural support for integrating low level knowledge with high level applications and user context.

Similar principles were used in [56] to build a distributed monitoring layer (called the awareness plane) that relies on autonomic computing principles. The proposed monitoring

framework consists of distributed traffic measurement and monitoring techniques, as well as dynamic configuration functionalities to perform traffic metering and collection and traffic data analysis. The use of semantically enriched models of the network and service infrastructure is also discussed in order to capture environmental constraints and dynamic resource usage.

**Multi-Agent Systems** Due to the existence of commonalities between agent-based approaches and *autonomic computing* principles, some research has focused on using multi-agent systems (MAS) theory to realise network self-management features. As explained in section 2.3.1, the self-management functionality in that case results from the interaction between management agents. Examples of approaches extending MAS principles for autonomic networks are [30][57][58][59][60]. In MAS-based approaches, a group of management agents embedded with specific management skills are coupled to the managed resources and a specific model is used to describe the relationships between the managed elements. Unlike IBM's view of autonomic components which considers that each managed resource is tightly coupled to a management entity, the link between the management entity and the managed resource is established in a flexible fashion based on the cooperation between management agents to decide which resource to control. As such, MAS provide a structured way to define agent interaction and collaboration and similar principles can be used to implement cooperating autonomic elements. A fundamental difficulty encountered in MAS-based approaches is to guarantee that the consistency between the decisions of different agents can be achieved so that a common global system goal can be satisfied.

**Control Theory Tools** A methodology based on control theory principles was used in [1] to design an autonomic component that can support network self-optimisation. In that work, the architecture of the autonomic component follows the model of a controlled system where the state of the system is incrementally adjusted according to environmental feedback so that desired objectives can be achieved. The model of a controlled system is depicted in figure (2.2).

In this model, the reference input represents the objective to achieve, the control input represents the system parameters controlled by the autonomic component and the transduced output represents the result of the reconfiguration on the system state. The three components of the model can be mapped to the self-management framework as follows. The system environment can be represented by the *target system* component that models the controlled system, the autonomic component can be represented by the *controller* component that exercises the control, and the monitoring interface can be mapped to the *transducer* component that translates the measured output so that it can be compared to the reference input or objective. Uncertainties in the environment and the target systems are represented by a disturbance input. The differ-

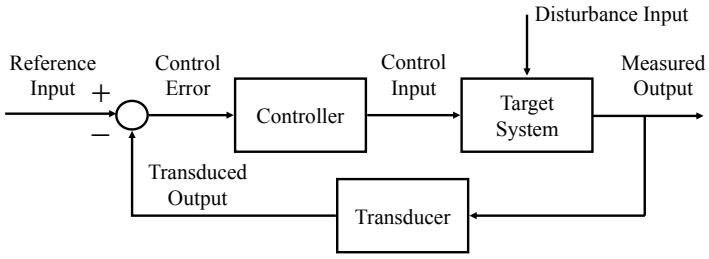



Figure 2.2: Elements of a control system [1].

ence between the measured output and the objective produces a control error that is provided as feedback to the controller.

Control theory was used in [1] to build a policy-based self-managing framework to regulate administrative utilities (e.g. backup utility) on an IBM database management system. More recently it was used in [61] to design an autonomic management control loop for IT service delivery in enterprise networks. A key feature that makes use of control theory attractive for the modelling of autonomic components is that it provides a set of tools and a methodology to analyse and design closed-loop functionality based on properties such as stability, accuracy, settling time and overshoot [1]. In order for such a methodology to be applied, however, a clear description of the system and its environment is necessary. While this can be straightforward in the case of isolated computing systems, it may be difficult for complex and large scale networks.

**Bio-inspired Solutions** Another research area that has received significant attention from the research community in the context of self-management, is that of bio-inspired networking solutions. Work in this area has investigated the application of design patterns inspired from biological processes such as diffusion, replication and stigmergy to describe the functionality and architecture of an autonomic component [44]. Biological processes exhibit, in particular, interesting properties for the design of decentralised decision-making processes.

An interesting example of network architecture designed according to biological features is the bio-inspired middleware platform presented in [62], which follows the model of a bee colony. In the proposed framework, network applications are built based on the combination of autonomic elements that represent different functionalities. Each autonomic element is defined as a *cyber-entity* that can perform different tasks such as communication, energy exchange and storage and pheromone emission. The organisation of the autonomic elements into different structures is realised through an interaction process that relies on two biological concepts: a) energy exchange, i.e. the autonomic entities can gain energy by providing services to each other,

and b) pheromone emission, i.e. the autonomic elements periodically emit some pheromone in the environment in order to indicate their presence and, as such, facilitate service discovery. In [63] a bio-inspired rerouting mechanism in case of link and node failure events was designed. The proposed algorithms rely on various biological processes such as the reaction-diffusion process that drives the self-organisation of biological cells, the chemotaxis process by which micro-organisms can direct their movements according to the concentration of certain chemicals in the environment, and the quorum sensing process that allows groups of cells to use information about their colony to initiate the production of specific types of substances. The reaction-diffusion and chemotaxis processes are used in particular to model the exchange of load information between neighbouring nodes while the quorum sensing process is used to describe how network nodes can cooperate for the purpose of traffic rerouting.

While bio-inspired solutions can provide useful tools for the development of distributed self-management approaches, these may not offer any guarantee in terms of scalability and as such, an analysis of the cost incurred in terms of communication overhead and convergence is required.

## 2.5 Adaptive Network Resource Management

Resource management in fixed networks is performed through traffic engineering functionality. Coupled to long term bandwidth provisioning, this aims at controlling and optimising routing configurations and traffic allocation in order to avoid the building of congestion [2][64], and, as such, to support good levels of quality of service. Current practices for managing network resources mainly rely upon offline traffic engineering approaches where the expected traffic demand is calculated from previous usage and specific routing configurations are produced to optimise resource usage over long timescales, e.g. weeks or months [65]. Routing configurations are typically computed by a central management system based on the estimation of the traffic demand for the next provisioning period.

Offline traffic engineering schemes have been extensively investigated both in the context of MPLS networks, by defining the Label Switched Paths (LSPs), and in the context of ordinary IP networks, by determining heuristics to tune the link weights that optimise some objective function, given a set of traffic matrices [66][67][68][69][70]. However, network traffic is known to be highly dynamic due to the variations in user demand, link failures, Border Gateway Protocol (BGP) rerouting or flash crowds [71]. Due to their static nature, offline approaches are unable to deal with dynamic conditions that cannot be reflected in the demand estimation and, as such, can well be sub-optimal in the face of changing or unpredicted traffic demand.

To overcome these limitations, online approaches that are able to react and adapt to current conditions in short timescales have been developed [72][73][74]. In contrast to offline schemes, online traffic engineering approaches do not rely on the knowledge of previous traffic characteristics to update the routing configurations. Instead, they dynamically adapt the settings in short timescales based on real-time information received from the network [65]. As such, online mechanisms aim at adaptively distributing the traffic load as evenly as possible onto network resources according to changing traffic conditions in order to accommodate unanticipated demand or traffic variations.

Driven by the rising cost of energy and increasing environmental consciousness, more recent research efforts have been focusing on the development of energy-saving resource optimisation techniques for green operations. It has been suggested that instead of statically provisioning the resources, a subset of elements (line cards, interfaces or even entire routers) could be reconfigured to enter a sleeping mode during off-peak times when the traffic load is low in order to save energy [75][76][77]. In order to guarantee global quality of service levels, mechanisms to select the network elements that can enter sleeping mode have been investigated. While some efforts (e.g. [77]) have considered time-driven approaches by which network configuration profiles can be computed *a priori* and applied according to the time period, others (e.g. [76]) have developed more adaptive approaches by which individual network devices can decide to enter sleep mode based on their awareness of current network conditions.

Another key resource management case study that has recently received a lot of interest concerns the management of caching infrastructures [78][79][80]. Previous activity in this domain has investigated quality of service-aware content location and delivery management methods, in particular, i.e. to control the placement of replicated content instances at different storage locations and to decide how to serve client requests for contents so that network resources can be better utilised.

The Ph.D. research presented in this thesis investigated key research issues associated with the three adaptive network resource management applications described in this section. In order to better highlight the contributions of this work, a detailed review of the relevant literature for each management application is provided in the subsequent chapters.

In order to be able to react to changing traffic and network state, adaptive resource management approaches need to be complemented by efficient monitoring mechanisms. Although the use of integrated monitoring functionality can be envisioned as a potential solution (e.g. [74][77]), this may be too restrictive, as the proposed solution may apply to a particular context only. Recent efforts have developed new protocols for distributed real-time monitoring that can

be used to support adaptive resource management schemes e.g. [81][82]. Furthermore, specific information management overlay structures have also been proposed [83][84].

## 2.6 Conclusion

This chapter introduced the state-of-the art in the area of autonomic computing and network self-management. In order to meet the requirements of emerging services, the future Internet will need to be flexible, reactive and adaptive. Network management functionality is essential in providing dynamic reactiveness and adaptability but current network management approaches have limitations, and are inadequate to meet the required demands. In search for a paradigm shift, research efforts have focused on self-management principles. These consist in embedding management intelligence into the network through in-network functionality that can adapt and react gracefully to changes based on feedback closed-loop control solutions. This will result in self-managed networks that can understand and analyse their environment and, as such, make optimal use of the resources as required by demanding applications. In order to enable self-managed applications and infrastructures, different high level architectures have been proposed in the literature. In particular, these define new interfaces between the management application and managed infrastructure, offering new models of abstraction to represent network operations and interaction processes. Previous research has, however, mainly focused on applying the proposed models to specific problems only, partially considering key research issues. This Ph.D. advances the state-of-the art by proposing a novel resource management framework to support self-management functionality, which is applied to three different resource management application scenarios.

## Chapter 3

# Management Substrate Structure for Dynamic Reconfiguration of Network Resources

## 3.1 Introduction

Network resource management approaches traditionally deployed by operators rely on offline functionality that can not easily deal with the traffic patterns of emerging services, which are becoming more dynamic and unpredictable. As such, solutions that are flexible and adaptive to traffic and network dynamics are of paramount importance. Furthermore, network resource management normally relies on centralised managers that periodically compute new configurations according to dynamic traffic behaviours. These centralised approaches have limitations especially in terms of scalability (i.e. communication overhead between the central manager and devices at runtime) and lag in the central manager reactions that may result in sub-optimal performance. To meet the requirements of emerging services, network resource management functionality that is decentralised, flexible, reactive and adaptive to traffic and network dynamics is necessary.

To overcome the limitations of current approaches, this chapter introduces a new in-network management framework for dynamic resource reconfiguration in fixed backbone networks. According to the proposed framework, the decision-making process is distributed across nodes in the network, so that each node is responsible for deciding on reconfiguration actions to take based on local feedback regarding the state of the network [5][6]. Nodes are equipped with the necessary logic that can allow them to perform reconfigurations, so that the network resources can be better utilised. In order to avoid inconsistencies between several independent decisions, the network nodes cooperatively decide on the most suitable changes to apply depending on network characteristics and conditions. The network nodes participating in the resource management process belong to a *management substrate* which is a logical structure used to facilitate the exchange of information between distributed decision-making points (for

management purposes). Due to the distributed nature of the decision-making process, the performance of the proposed management scheme in terms of communication overhead can be affected by the structure of the management substrate. Three different topology structures that connect management substrate nodes are investigated in this chapter. A set of methods to compute the proposed topology structures is described and a quantitative and qualitative comparison of the three topologies according to different parameters is presented.

The rest of this chapter is organised as follows. Related work is presented in section 3.2. Section 3.3 introduces in more detail the in-network management substrate framework developed to perform adaptive resource management. Sections 3.4, 3.5 and 3.6 present the three different topology structures. The characteristics of the different structures are evaluated and discussed in section 3.7. The contributions of this work are finally summarised in section 3.8.

## 3.2 Related Work

Interaction and communication between autonomic elements have been described as fundamental architectural features of autonomic computing systems in [32]. The authors highlight in particular that autonomic elements can establish relationships between each other in order to request or offer some service. In that work, the authors propose a generic model based on negotiation to drive the interaction between autonomic elements. Other generic interaction models have been considered in [42], where four types of behaviour that can be exhibited by an autonomic element towards other autonomic elements are described, i.e. cooperative, selfish, punishment and mixed behaviour. Communication models between network entities to support management tasks have also been considered by the authors in [85] in the context of autonomic networks. In [85], the interaction between the decision elements relies on a hierarchical structure in which the decisions taken by each decision element are orchestrated by one or more "arbiter" elements that are in charge of detecting potential overlapping or contradicting actions and configurations.

Some research efforts have also investigated the use of generic hierarchical architectures inspired by multi-agent systems to support interaction and cooperation between nodes [30][57]. The use of gossip-based protocols to distribute information across distributed decision-making points was considered in [86][87][88]. According to gossip-based approaches, the interaction between nodes relies on a random process, so that at regular time intervals, one node in the network initiates a communication with a randomly selected neighbour in order to exchange information. The work in [86] focused on the development of scalable and adaptive mechanisms

for calculating aggregates in a pro-active manner. A gossip-based approach was used in [87] for dynamic resource allocation in cloud environments and in [88] for development of decentralised self-adaptive aggregation mechanisms.

The design of logical infrastructures to connect a set of nodes has received a lot of attention from the research community over the last decade, especially in the context of overlay networks, e.g. [89][90][91][92]. While research efforts in this area have focused on developing scalable systems through optimised logical topologies and overlay routing protocols, the purpose of the work presented in this chapter is not to investigate features and techniques to support overlay systems. It focuses, instead, on the design of topology structures that can offer good performance in terms of communication cost and management overhead for supporting the interaction between network reconfiguration entities.

## 3.3 In-Network Management Substrate

### 3.3.1 Notations and Definitions

The following notations are used in this chapter. Let $\mathcal{L}$ be the set of network links and $\mathcal{N}$ the set of network nodes. The latter is further divided into the set of network edge nodes, i.e. network nodes generating and absorbing traffic, and the set of network core nodes. Each node has a unique identifier. Unless otherwise stated, the network infrastructures considered in this thesis correspond to intra-domain fixed backbone networks.

### 3.3.2 Decentralised Resource Management Framework

In the proposed in-network resource management framework, network edge nodes are embedded with a level of intelligence that allows them to react to network conditions in a decentralised and adaptive fashion based on periodic feedback information received from the network. Compared to centralised offline solutions, where reconfigurations are decided by a centralised management system that has global knowledge of the network, reconfiguration decisions are directly taken by the network edge nodes that coordinate among themselves in order to decide upon the best sequence of actions to perform to satisfy a common objective. These can, for instance, consist of adjustments to routing parameters for load balancing purposes. In order to support this decentralised decision-making process, the network edge nodes are organised into a *management substrate*, which is a logical structure used to facilitate the exchange of information between decision-making entities. The management substrate is used by the edge nodes for coordination purposes, in particular, since it provides a means through which nodes can communicate. It is worth mentioning that the substrate is only used for signalling, and not for direct traffic routing/forwarding.

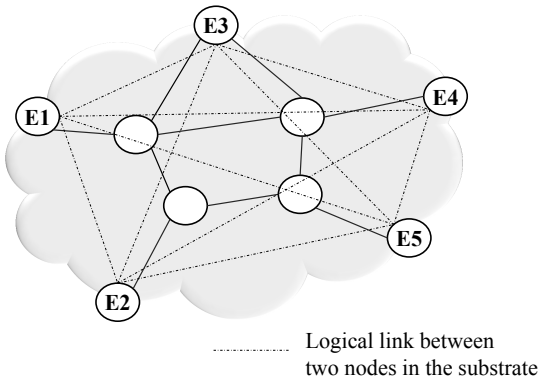



Figure 3.1: In-network management substrate overview.

A management substrate structure example is depicted in figure (3.1), where each network edge node $E$ is logically connected to a set of other network edge nodes (neighbour nodes in the management substrate ($MS$)). Any $MS$ node can directly communicate only with its neighbours, which are defined by the topological structure used. The choice of the substrate topology can be driven by different parameters related to the physical network, such as its topology, the number of edge nodes, but also by the constraints of the coordination mechanism between the nodes and the associated communication protocol. The overhead incurred by the communication protocol in terms of delay and number of messages exchanged, for example, is a key factor that can influence the choice of the topology.

In the proposed management framework, it is assumed that all $MS$ nodes are network edge nodes. As such, unless otherwise stated, the term node is used to refer to a network edge node. In addition, this work assumes that the nodes do not fail.

### 3.3.3 Substrate Characteristics

Three different topology structures to connect the $MS$ nodes are investigated. The proposed structures differ in terms of degree of connectivity (i.e. number of neighbours of each node in the substrate) and the number of hierarchy levels. The degree of connectivity of a topology defines the visibility of each node in the substrate and can thus affect the volume of information that needs to be maintained at each $MS$ node. In addition, the number of hierarchy levels in the structure drives the number of modes of communication required between $MS$ nodes, which may influence the complexity of the communication protocol. The main objective considered for the design each structure is to minimise the communication overhead incurred by the coordination process. This is defined by the volume of signalling messages and the delay, which is driven by the communication cost between $MS$ nodes.

The communication cost between two $MS$ nodes $i$ and $j$ is defined as the cost of the

logical link between the two nodes. This cost, denoted $C_{\mathrm{LL}}(ij)$, is driven by the cost of the path between node $i$ and node $j$ in the underlying physical network topology, where the cost of a path is equal to the sum of the cost of the links involved in the path, i.e.:

$$C_{\mathrm{LL}}(ij) = \sum_{l \in \mathcal{L}} \delta_{ij}^{l} \cdot c(l) \tag{3.1}$$

where $\delta_{ij}^{l}$ is a $\{0-1\}$ binary variable equal to 1 if link $l$ is included in the path between nodes $i$ and $j$, and $c(l)$ is the cost of link $l$.

The cost $c(l)$ of link $l$ can be defined, for instance, according to the administrative cost (i.e. link weight) which is the metric used to compute the shortest paths. Administrative costs are usually assigned based on the characteristics of the underlying physical network topology and on traffic engineering requirements. A common practice is to set link weights equal to the inverse of the link capacities [93]. These costs may not, however, be sufficient to account for the communication cost in terms of delay between two nodes since the delay is also influenced by the geographical distance between the nodes (i.e. propagation delay). In order to take the geographical distance into account, an additional metric, called link distance factor ($c_{\varphi}$), is defined for each link. This represents the relative distance between two nodes in the network and is defined as the ratio between the geographical distance $d_l$ (e.g. in kilometres) obtained for each link $l$ divided by the smallest geographical distance observed in the network, i.e.

$$c_{\varphi}(l) = \frac{d_l}{\min_{l \in \mathcal{L}}(d_l)} \tag{3.2}$$

The cost of a link $l$ is then defined as the product of the link administrative cost $c_{\alpha}(l)$ and the link distance factor $c_{\varphi}(l)$, i.e.

$$c(l) = c_{\alpha}(l) \cdot c_{\varphi}(l) \tag{3.3}$$

Administrative costs are usually positive numbers and, as such, the cost $c(l)$ is, in general, a non-negative real number. In addition, it should be noted that other metrics, such as the round-trip time variability or link load, could also be considered to represent the communication cost in terms of delay between two nodes. These are dynamic factors that depend on the network conditions. In contrast, the cost $c(l)$ is independent of current conditions. This can be computed in an offline fashion based on the values of $c_{\alpha}(l)$ and $c_{\varphi}(l)$, and, as such, is more appropriate for the construction of the management substrate.

It is assumed that the path used between two nodes is the shortest-path and that all network links are bidirectional, so that for any pair of nodes $i$ and $j$, $C_{LL}(ij) = C_{LL}(ji)$. Finally, it is worth noting that reliable transfer of messages between nodes is assumed and that the proposed

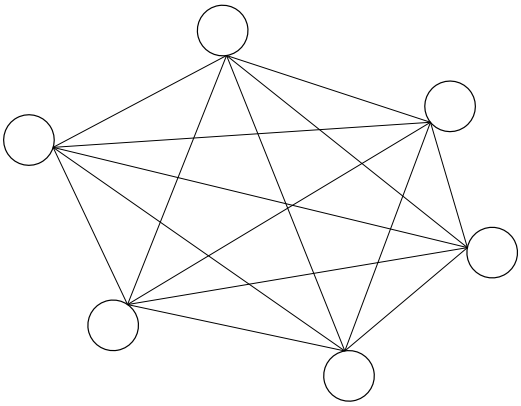


Figure 3.2: Full-mesh topology structure.

framework relies on the Transmission Control Protocol (TCP) [94] as the underlying transport protocol.

## 3.4 Full-mesh Management Substrate Structure

### 3.4.1 Topology Structure

$MS$ nodes are connected according to a full-mesh topology as shown in figure (3.2), where each node is logically connected to every other node.

The full-mesh model is a flat structure (i.e. not hierarchical) and has the maximum degree of connectivity. The total number of logical links is equal to $\frac{N \cdot (N-1)}{2}$, where $N$ is the total number of $MS$ nodes. In this model, each $MS$ node has a global view about other $MS$ nodes. This provides the maximum flexibility in the choice of neighbours with which to communicate since all $MS$ nodes belong to the set of neighbours. Each $MS$ node, however, needs to locally maintain information about every other $MS$ node, which may causes problems of scale with respect to information storage with an increasing number of substrate nodes.

### 3.4.2 Communication Protocol For Management Operations

**Communication Model** As explained in section 3.3.2, the $MS$ facilitates the communication between reconfiguration entities (i.e. edge nodes). These can, for instance, exchange information related to reconfigurations to perform or request assistance from each other when local reconfigurations are not possible. The communication model used in the full-mesh topology follows a star structure centred on the node that initiates the communication as depicted in figure (3.3). The initiator (represented as a grey disc in the figure) sends a $REQUEST$ message to all its neighbours in the management substrate. The $REQUEST$ message can be used, for instance, to request some information from the other nodes in the substrate. Upon receiving a $REQUEST$ message, each neighbour node analyses the content of the message to decide whether it can provide a satisfactory reply to the request. In case it can, the node appends the

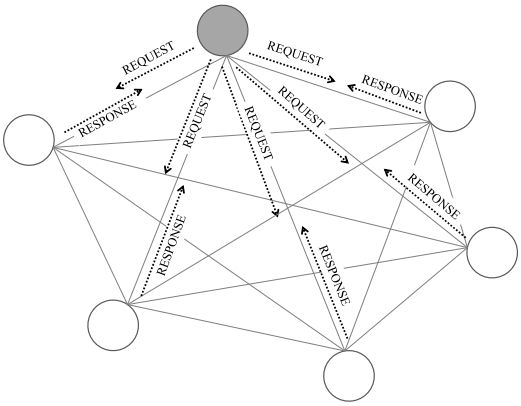



Figure 3.3: Communication model in the full-mesh topology structure.

required information to a $RESPONSE$ message and forwards it back to the initiator. Otherwise, a negative $RESPONSE$ message is returned. A similar communication protocol can also be used to notify other nodes in the substrate about local changes. In order to minimise the communication overhead due to signalling, the size of the messages needs to be small.

**Communication Overhead** Intuitively, the total number of messages sent by the initiator node is equal to the number of neighbours it has, i.e. $(N-1)$ where $N$ is the number of nodes in the management substrate. The volume of signalling messages increases linearly with the number of $MS$ nodes. Due to the structure of the full-mesh topology, the communication cost in terms of delay mainly depends on the maximum round-trip time (RTT) between the initiator and any other $MS$ node. As shown in previous studies [95][96], the values of the RTT are mainly influenced by the geographical distribution of network nodes. It can therefore be inferred that the communication cost in terms of delay in the full-mesh model is driven by the maximum geographical distance between $MS$ nodes.

## 3.5 Ring Management Substrate Structure

### 3.5.1 Topology Structure

$MS$ nodes are connected according to a ring topology as shown in figure (3.4), where each node is logically connected to two other nodes only.

The ring topology is also a flat structure. Unlike the full-mesh topology, however, it has a low degree of connectivity. The total number of logical links is equal to the number of nodes in the substrate. The view of each $MS$ node is limited to its two direct neighbours only, and it is thus not possible to directly communicate with any other nodes in the substrate. In order to communicate, a message needs to be sent over the ring until it reaches its destination. Given that the total communication cost (i.e. delay) can be defined as the sum of the cost between all successive nodes, it is affected by the order according to which the nodes are connected. Based

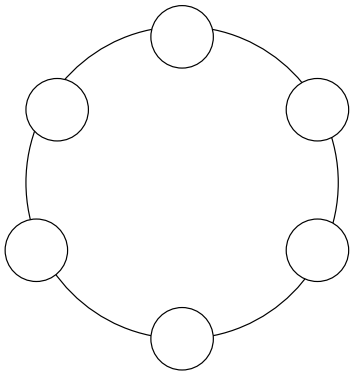


Figure 3.4: Ring topology structure.

on the definition of the cost provided in section 3.3.3, the next subsection presents a heuristic to connect a set of nodes according to a ring topology, so that the total cost (i.e. sum of the cost between each successive node) is minimised.

### 3.5.2 Ring Model Construction

This problem is similar to the Travelling Salesman Problem (TSP) [97]. The TSP is a well-known NP-Hard combinatorial optimisation problem that consists in determining, given a list of locations and their pairwise distances, the shortest possible route that visits each location exactly once and that returns to the starting location. Although a number of approaches with near-optimal performance exist in the literature to solve the TSP, they are computationally expensive. In order to keep the complexity of the construction algorithm low, an approach based on the simple *Nearest Neighbours* tour construction heuristic [98] was developed. This has a time complexity $O(N^2)$, with $N$ being the number of nodes to consider.

The principle of the proposed approach is as follows. Given a node $i$, node $j$ is selected as the successor of $i$ such that the cost $C_{\text{LL}}\,(ij)$ is the lowest. The *Nearest Neighbour* algorithm considers each node $i$ iteratively and selects, among other $MS$ nodes that have not already been considered, the successor of $i$, i.e. the node with the lowest logical link cost to $i$. The algorithm terminates when all nodes have been considered and the successor of the last node is set to be the initial node.

### 3.5.3 Communication Protocol For Management Operations

**Communication Model** The communication between $MS$ nodes in the ring model relies on a hop-by-hop mechanism as depicted in figure (3.5). Communication is unidirectional, which means that a node can only pass information to its immediate neighbour in the ring. To communicate with any other node a message needs to be sent over the ring until it reaches its destination. To satisfy the one path message transfer procedure, the communication direction

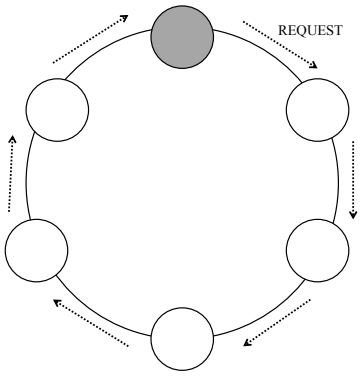



Figure 3.5: Communication model in the ring topology structure.

followed in the ring must be fixed but it can be either anticlockwise or clockwise.

The initiator node (represented as a grey disc in the figure) sends a $REQUEST$ message to one of its neighbouring nodes according to the communication direction followed. The initiator node then enters a listening period where it waits for the message to travel hop-by-hop through the ring until it reaches the initiator again. Upon receiving the $REQUEST$ message, the next hop node analyses the content of the message, appends the required information as well as its identity to the message and forwards it to the next hop node.

**Communication Overhead** In the case of the ring model, the number of neighbours of each $MS$ node is independent of the total number of nodes in the substrate. In order to communicate, the initiator node sends a message to one of its direct neighbours and the message is forwarded hop-by-hop to the other nodes in the ring. As such, the number of messages sent by each $MS$ node is independent of the total number of nodes in the substrate. In contrast, due to the characteristics of the communication model, it can be inferred that the communication cost in terms of delay will be driven by the number of nodes in the ring, and as such, there may be some scalability limitations as the number of nodes in the substrate increases.

## 3.6 Hybrid Management Substrate Structure

### 3.6.1 Topology Structure

Due to their characteristics, the full-mesh and ring models present some scalability limitations when the number of $MS$ nodes increases. In the case of the full-mesh, this can incur a significant increase in the volume of substrate information to be maintained locally at each $MS$ node and in the case of the ring model it can significantly affect the total communication delay. In order to overcome the limitations of these two simple structures, the design of a more sophisticated model to organise the $MS$ nodes is investigated. This model, referred to as a hybrid topology, is a combination of the ring and full-mesh structures, as depicted in figure (3.6).

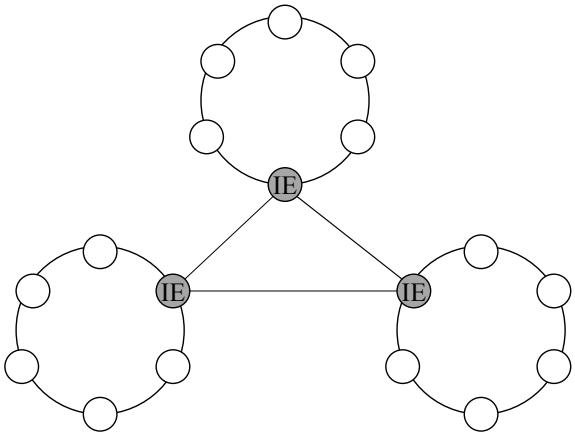



Figure 3.6: Hybrid topology structure.

The hybrid topology consists of a set of rings inter-connected in a fully-meshed fashion through *Intermediate Entity* ($IE$) nodes, so that there exists exactly one $IE$ node in each ring. More specifically, $MS$ nodes are partitioned into at least two clusters, so that nodes in each of the clusters are connected according to a ring topology. One node is then selected in each cluster to be the $IE$, i.e. to act as an interface to the other clusters. It is worth noting that each $MS$ node belongs to one cluster only. One of the incentives for using a hybrid structure is to provide a trade-off in terms of performance between the message overhead and the delay incurred when two $MS$ nodes need to exchange information. Such a trade-off raises some requirements when deciding how to connect nodes according to the hybrid model. The next section presents a set of methods that define how to partition the $MS$ nodes into clusters and how to select the $IE$ node in each cluster.

### 3.6.2 Hybrid Model Construction

**Constructing Multiple Rings** One key challenge when forming the hybrid structure is to determine which metric to use in order to partition the $MS$ nodes into clusters. A natural choice is to use the logical link cost metric defined in section 3.3.3, which is a function of the link administrative cost and the geographical distance. As such, nodes are clustered based on their proximity with respect to the logical link cost.

In order to reduce the communication delay compared to the ring structure, the total communication cost permitted in each sub-ring of the hybrid structure needs to be less than an upper bound threshold $\theta$. The value of the threshold is a key factor since it can influence whether a node should be considered as a member of a specific ring or not, and, as such, directly affects the size of each sub-ring. To set the appropriate threshold value, it is also essential to take into account the fact that nodes located in different sub-rings can communicate. Given that nodes

can directly communicate between each other in the full-mesh model, it can be inferred that the communication cost in this model is less than the cost in the ring model. As explained in section 3.4.2, the communication cost of the full-mesh structure is driven by the maximum geographical distance between $MS$ nodes and, as such, by the maximum logical link cost in the $MS$. This can therefore be used as a reference metric to derive the value of the threshold to apply to the total cost in each sub-ring. Two cases are investigated:

1. $\theta$ is equal to $\theta_{\text{HalfMax}}$, i.e. to half of the maximum logical link cost obtained if $MS$ nodes were connected in a full-mesh fashion.

2. $\theta$ is equal to $\theta_{\text{Avg}}$, i.e. to the average logical link cost obtained between all possible pairs of nodes if $MS$ nodes were connected in a full-mesh fashion.

An approach was designed to partition the $MS$ nodes into the different clusters according to $\theta$, and compute the resulting sub-rings. The proposed algorithm follows an iterative process where all $MS$ nodes are considered one-by-one. The number of clusters is not determined a priori. One cluster is initially formed by the algorithm and nodes are successively added to this cluster until the threshold condition is violated. In this case, the initial cluster is said to be complete and a new cluster is formed to accommodate the remaining nodes. The different clusters are thus formed successively according to the threshold value $\theta$ and the order in which nodes are considered. To ensure that each node belongs to one cluster only, the algorithm maintains the list of $MS$ nodes that have not been considered yet. The list contains initially all $MS$ nodes and is updated at each iteration by removing the node selected by the algorithm. The output of the algorithm is a set of rings.

The steps of the algorithm are presented in Algorithm (3.6.1). $N_{curr}$ is the node considered by the algorithm at each iteration and $\mathcal{N}_{\text{wait}}$ is the list of the $MS$ nodes that have not been considered yet. $N_{\text{ini}}$ is the initial node and $\mathcal{S}_{\text{RINGS}}$ is the set of constructed rings.

Two criteria to select the initial node are compared. In the first case, the node connected to the logical link with the lowest cost in the $MS$ is selected as the initial node, while, in the second case, the node connected to the logical link with the highest cost in the $MS$ is selected. Given that logical links are bidirectional, the node with the lowest identifier is selected by default.

There may be cases where some of the sub-rings obtained contain one element only, which is not consistent with the definition of the hybrid topology. The structure of each sub-ring is therefore analysed at the end of the algorithm. If single node sub-rings are found, the algorithm disregards them and assigns the involved nodes to other sub-rings, so that the selected rings are those for which the addition of an extra node leads to the lowest increase in terms of cost.

**Algorithm 3.6.1** Main steps of the Multiple Rings Construction Algorithm.

1. Select an initial node $N_{\text{ini}}$, set $N_{\text{curr}}$ to $N_{\text{ini}}$ and remove $N_{\text{ini}}$ from $\mathcal{N}_{\text{wait}}$.
2. Create a new cluster $\mathcal{C}$ with $N_{\text{curr}}$.
3. Compare the cost of the logical links from $N_{\text{curr}}$ to all nodes in $\mathcal{N}_{\text{wait}}$. Select the pair (i.e. logical link) with the lowest cost and mark the relevant peer node as $N_{\text{test}}$.
4. Apply the ring construction algorithm described in section 3.5.2 to the set of nodes formed by the union of the set of nodes in cluster $\mathcal{C}$ and $N_{\text{test}}$. Determine the total ring cost $C_{\text{ring}}$.
5. Compare $C_{\text{ring}}$ to the threshold value $\theta$. If $C_{\text{ring}} \leq \theta$, add $N_{\text{test}}$ to cluster $\mathcal{C}$, remove $N_{\text{test}}$ from $\mathcal{N}_{\text{wait}}$ and set $N_{\text{curr}}$ to $N_{\text{test}}$. Go back to step 3. If $C_{\text{ring}} > \theta$, apply the ring construction algorithm to nodes in cluster $\mathcal{C}$ and add the resulting ring to $\mathcal{S}_{\text{RINGS}}$. Set $N_{\text{curr}}$ to $N_{\text{test}}$, remove $N_{\text{test}}$ from $\mathcal{N}_{\text{wait}}$ and go back to step 2.
6. Continue until $\mathcal{N}_{\text{wait}}$ is empty.

**Intermediate Entity Selection** Another key issue raised by the design of the hybrid topology is the selection of the most appropriate $IE$ in each sub-ring, so that these can be efficiently interconnected in a full-mesh. In a similar fashion to the method used to select successor nodes in the ring construction algorithm, $IE$ nodes are chosen according to their proximity, in terms of logical link cost, to other rings. This can be formally described as the problem to determine which node to select in each sub-ring so that the maximum logical link cost between all pairs of the IE nodes is minimised. In order to simplify the Intermediate Entity Selection procedure, a heuristic to select the node in each sub-ring that is the closest on average to every other remote node in the substrate (i.e. to the nodes in other sub-rings) was investigated. The proposed approach relies on an iterative process, where sub-rings are considered one-by-one, so that at each iteration, one node in the considered sub-ring is selected as the $IE$. In order to select the appropriate $IE$ in each ring, the algorithm computes, for each node in the ring, the average logical link cost to every other remote node. The selected $IE$ in each ring is the one with the lowest average cost, i.e. the node that is closest on average to every other node in the substrate.

### 3.6.3 Communication Protocol For Management Operations

**Communication Model** The protocol for the communication between the $MS$ nodes organised into a hybrid structure supports two modes of communication as depicted in figure (3.7) and described below.

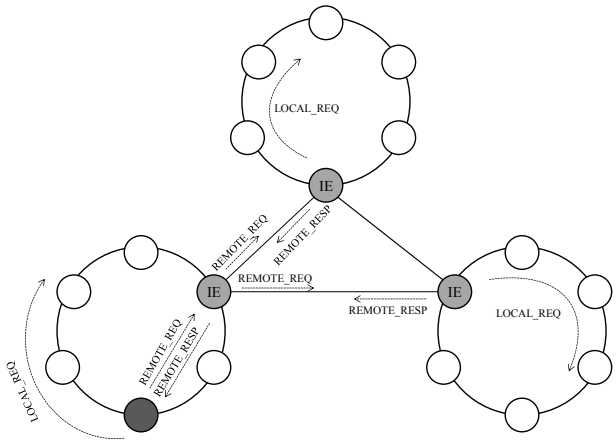



Figure 3.7: Overview of the hybrid topology communication model.

**Local Sub-ring Communication:** The first mode concerns the communication between nodes located in the same ring. This mode corresponds to the case where the node that initiates the communication (represented by a grey disc in the figure) needs to exchange some information with another (other) node(s) in the local sub-ring. In that case, the hop-by-hop mechanism described in section 3.5.3 for the ring model is used. More specifically, the initiator node sends a local request in the form of a $LOCAL\_REQ$ message to one of its neighbouring nodes according to the communication direction followed. The message then travels hop-by-hop through the ring until it reaches the initiator node again.

**Remote Sub-ring Communication:** The second mode concerns the communication between nodes located in different rings. This can be triggered, for example, in case the information required by the initiator node cannot be found in the local sub-ring, and, as such, needs to be retrieved from a node located in a remote sub-ring. To do this, the initiator needs to first communicate with its local $IE$ since this node acts as an interface to the other rings. It is assumed that the address of the $IE$ in a given sub-ring is known by all the nodes of that ring (this can be pre-configured during the setup phase of the network). The initiator starts by sending a remote request ($REMOTE\_REQ$) message directly to its $IE$ node, which then forwards it to all the other $IE$ nodes of the $MS$. Each $IE$ is subsequently responsible for circulating a $LOCAL\_REQ$ message in its local ring. Upon receiving this message back, each $IE$ analyses its content and creates a remote response ($REMOTE\_RESP$) message that contains information about potential satisfactory replies from its ring. This is sent back to the original requesting $IE$, which forwards it to the initiator.

**Communication Overhead** In the full-mesh $MS$ topology model, the communication overhead incurred when a node requests information is proportional to the number of nodes in the $MS$, since a message is exchanged with every other node (section 3.4.2). According to the

communication protocol used in the hybrid model, the total number of messages exchanged depends on the communication mode considered. For the local sub-ring communication case, only one message needs to be sent by each node: a $LOCAL\_REQ$ message to its direct neighbour. For the remote sub-ring communication case, however, with $r$ being the number of sub-rings, one $REMOTE\_REQ$ message is sent by the initiator node to the local $IE$ and $(r-1)$ $REMOTE\_REQ$ messages are sent from the local $IE$ to other $IE$ nodes in the substrate. As such the communication overhead in terms of number of messages in the hybrid model is, in the worst case, proportional to the number of sub-rings. Compared to the full-mesh topology, the performance of the hybrid model improves as the size of each ring increases, and consequently, as the number of rings decreases.

Given the hybrid nature of the model, it can be deduced that the communication cost in terms of delay will be driven by the characteristics of the full-mesh and ring structures. It can therefore be inferred that the total delay will be influenced by the size of the largest sub-ring and the maximum distance between $IE$ nodes. In addition, it is expected that better performance in terms communication cost will be achieved with the hybrid model than with the ring model but that these may not outperform the performance obtained with the full-mesh approach. An quantitative evaluation of the communication cost obtained with each topology model is presented in section 3.7.3.

## 3.7 Implementation and Evaluation

### 3.7.1 Software Architecture

To evaluate the proposed topology models, a Java program that computes the $MS$ topology structure corresponding to any physical network topology was developed. The program takes as input the network topology, the identifiers of the network edge nodes and a set of configuration parameters. The latter allow the user to control the type of $MS$ structure to compute (ring, full-mesh, hybrid), the logical link cost model, the threshold value and the initial node selection criterion. An overview of the main components of the program is depicted in figure (3.8). In order to guarantee the integrity of the results, testing procedures were applied to the different methods and classes. The program was run on a laptop with 2.80 GHz Intel Core i7-2640M processor and 8 GB memory.

### 3.7.2 Management Substrate Construction Algorithms Performance Evaluation

**Experiment Settings** The impact of key parameters associated with the construction process was evaluated and analysed using two real PoP (Point of Presence)-level network topologies, Abilene [99] and GEANT [100]. The PoPs, in each topology, are mapped to cities, which

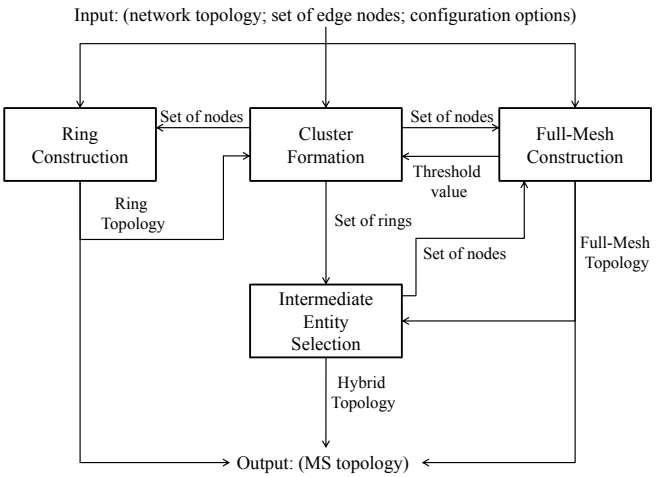



Figure 3.8: Overview of the Management Substrate computation software architecture.

enables the determination of geographical distance between every pair of nodes (Google Maps was used). While the full 11-node topology is used in the case of the Abilene network, a reduced GEANT topology, which excludes the two non-European PoPs, i.e. 21 instead of 23 nodes, is considered.

**Ring Construction** To evaluate the performance of the ring construction algorithm described in section 3.5.2, the total ring cost obtained for a set of nodes is compared to the cost of the optimal ring structure [97]. The optimal ring structure is computed using the *glpsol* GLPK (GNU Linear Programming Kit) linear programming/mixed integer programming solver [101]. This is used in the reference traffic engineering tool TOTEM (TOolbox for Traffic Engineering Methods) [102][103]. The performance obtained by a method that randomly connects the nodes in a ring is also considered. To analyse the influence of the number of nodes on the performance of the algorithm, experiments using different number of nodes in both the Abilene and GEANT networks are performed. A subset of nodes is randomly selected and connected into a ring according to the three approaches considered, i.e. the proposed algorithm, the optimum and the random approach. To obtain a better approximation of the cost in the random case, the number of executions of the algorithm is proportional to the number of nodes and the results are averaged. The logical link cost is equal to the product of the administrative link weight and the geographical distance.

The deviation of the total ring cost obtained with the proposed and the random algorithms from the optimum, for different number of nodes, is depicted in figure (3.9). The deviation increases linearly with the number of nodes for both algorithms. Given that the proposed approach follows an iterative process where nodes are iteratively added to the ring structure, the error made at each iteration in the choice of a successor node incurs a cost penalty to the total

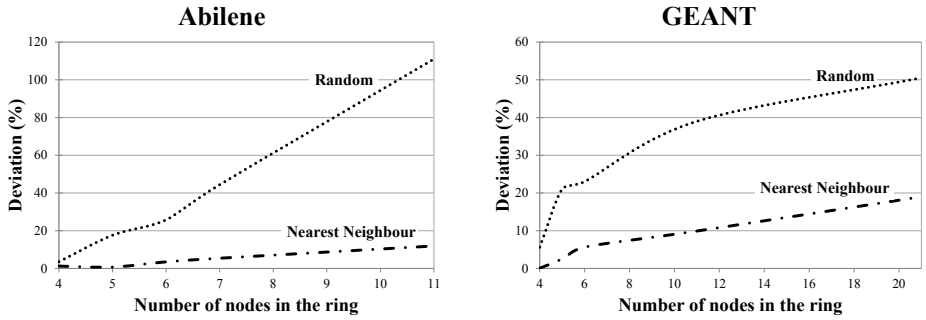



Figure 3.9: Evolution of the deviation from the optimum in case of the Abilene and GEANT networks.

ring cost. The penalty increases as the number of nodes to consider increases, and, as a result, the deviation from the optimum increases. It can be noticed, however, that the proposed algorithm outperforms the random one since the deviation is significantly lower in all cases.

**Multiple Rings Construction** This subsection provides an analysis of how the logical link cost $C_{\mathrm{LL}}$, the threshold $\theta$ and the initial node selection criterion can influence the structure of the sub-rings (i.e. number and size) computed according to the algorithm described in section 3.6.2. A comparison of the structures obtained when using the threshold values $\theta_{\mathrm{HalfMax}}$ and $\theta_{\mathrm{Avg}}$, and the initial node selection criterion lowest $C_{\mathrm{LL}}$ and highest $C_{\mathrm{LL}}$, as explained in section 3.6.2, is performed. In addition, two different cases for the logical link cost are considered: a) the administrative link weight of each network link is set to 1 (i.e. in this case, the $C_{\mathrm{LL}}$ is mainly driven by the geographical distance between the nodes), and, b) the administrative link weights are the original ones. These are denoted as $Model_{1\times\mathrm{D}}$ and $Model_{\mathrm{W}\times\mathrm{D}}$, respectively. The sub-ring structures obtained in the 8 different scenarios in both the 11-node Abilene network and the 21-node GEANT network are analysed.

To compare the different scenarios, three metrics are defined to describe the characteristics in terms of size of the different sub-rings obtained in each scenario. The $MinDeviation$ metric is equal to the ratio of the minimum total ring cost among the different sub-rings to the threshold value $\theta$. In a similar fashion, the $MaxDeviation$ metric is defined as the ratio of the maximum total ring cost to the threshold value $\theta$. Finally the $AvgDeviation$ metric is defined as the ratio of the average total ring cost between the different sub-rings to the threshold value $\theta$. The results obtained for each scenario in the case of Abilene and GEANT are reported in Table (3.1).

As observed, using $Model_{1\times\mathrm{D}}$ to set the logical link cost leads on average to the formation of more sub-rings in both Abilene and GEANT, although the difference is smallest in the Abilene case given the small size of the network. In the case of GEANT, it can be noticed that using the threshold value $\theta_{\mathrm{HalfMax}}$ results in sub-rings that are more balanced in terms of

| Cost $C_{LL}$ | Threshold $\theta$ | Initial Node | Rings Size | Min Deviation | Max Deviation | Avg Deviation |
|---|---|---|---|---|---|---|
| **GEANT Topology** | | | | | | |
| $Model_{1\times D}$ | $\theta_{Avg}$ | Lowest | 4,6,3,2,2,2,2 | 0.71 | 1.92 | 1.20 |
| $Model_{1\times D}$ | $\theta_{Avg}$ | Highest | 5,2,3,3,3,3,2 | 0.53 | 2.47 | 1.16 |
| $Model_{1\times D}$ | $\theta_{HalfMax}$ | Lowest | 4,6,3,2,2,2,2 | 0.60 | 1.62 | 1.01 |
| $Model_{1\times D}$ | $\theta_{HalfMax}$ | Highest | 5,2,2,3,2,5,2 | 0.56 | 2.08 | 0.90 |
| $Model_{W\times D}$ | $\theta_{Avg}$ | Lowest | 9,6,3,3 | 1.00 | 8.98 | 4.53 |
| $Model_{W\times D}$ | $\theta_{Avg}$ | Highest | 2,5,7,5,2 | 0.80 | 8.10 | 3.61 |
| $Model_{W\times D}$ | $\theta_{HalfMax}$ | Lowest | 15,6 | 2.39 | 3.01 | 2.70 |
| $Model_{W\times D}$ | $\theta_{HalfMax}$ | Highest | 15,6 | 2.39 | 3.01 | 2.70 |
| **Abilene Topology** | | | | | | |
| $Model_{1\times D}$ | $\theta_{Avg}$ | Lowest | 3,2,3,3 | 0.28 | 1.68 | 1.05 |
| $Model_{1\times D}$ | $\theta_{Avg}$ | Highest | 3,2,3,3 | 0.67 | 0.96 | 0.82 |
| $Model_{1\times D}$ | $\theta_{HalfMax}$ | Lowest | 3,2,3,3 | 0.27 | 1.60 | 1.00 |
| $Model_{1\times D}$ | $\theta_{HalfMax}$ | Highest | 3,2,2,2,2 | 0.22 | 1.36 | 0.71 |
| $Model_{W\times D}$ | $\theta_{Avg}$ | Lowest | 5,3,3 | 0.78 | 1.90 | 1.26 |
| $Model_{W\times D}$ | $\theta_{Avg}$ | Highest | 3,2,3,3 | 0.78 | 1.12 | 0.95 |
| $Model_{W\times D}$ | $\theta_{HalfMax}$ | Lowest | 5,3,3 | 0.67 | 1.63 | 1.09 |
| $Model_{W\times D}$ | $\theta_{HalfMax}$ | Highest | 3,2,3,3 | 0.67 | 0.96 | 0.82 |

Table 3.1: Multiple Rings Evaluation.

size, especially when $Model_{W\times D}$ is used. More precisely, when comparing the ratio between $\theta_{HalfMax}$ and $\theta_{Avg}$ in the case of $Model_{W\times D}$, it can be observed that its value is around 3 for GEANT, whereas it is equal to 1.15 for Abilene. As a result, the structure of the sub-rings obtained are strongly affected by the value of $\theta$ in the case of GEANT. In the case of $Model_{1\times D}$, the ratios are 1.18 and 1.04 for GEANT and Abilene, respectively, and as shown in the table, the structure of sub-rings is less affected by the choice of the threshold value. In addition, it can be noticed that the structure of the sub-rings is not significantly affected by the initial node selection criterion in all the cases.

**Intermediate Entity Selection** An analysis of how the logical link cost $C_{LL}$ and the threshold $\theta$ can influence the intermediate entity selection in each sub-ring is finally presented in Table (3.2). Here, the highest $C_{LL}$ is used as the initial node selection criterion. In a similar fashion to previous experiments, a metric is defined to compare the different scenarios. This, denoted $Deviation$, represents the ratio of the maximum logical link cost between the different $IE$ nodes to the value of the average ring cost obtained in the corresponding scenario in Table (3.1). As observed, the value of $Deviation$ increases with the number of sub-rings. A larger number of sub-rings means that more clusters were formed during the multiple rings construc-

| Cost $C_{LL}$ | Threshold $\theta$ | Rings Size | Selected IE Nodes | Deviation |
|---|---|---|---|---|
| **GEANT Topology** | | | | |
| $Model_{1\times D}$ | $\theta_{Avg}$ | 5,2,3,3,3,3,2 | 16,21,1,20,17,7,2 | 1.310 |
| $Model_{1\times D}$ | $\theta_{HalfMax}$ | 5,2,2,3,2,5,2 | 10,16,21,1,20,17,2 | 1.28 |
| $Model_{W\times D}$ | $\theta_{Avg}$ | 2,5,7,5,2 | 21,17,1,7,20 | 0.120 |
| $Model_{W\times D}$ | $\theta_{HalfMax}$ | 15,6 | 9,8 | 0.013 |
| **Abilene Topology** | | | | |
| $Model_{1\times D}$ | $\theta_{Avg}$ | 3,2,2,2,2 | 2,6,11,7,10 | 2.753 |
| $Model_{1\times D}$ | $\theta_{HalfMax}$ | 3,2,2,2,2 | 2,6,11,7,10 | 2.753 |
| $Model_{W\times D}$ | $\theta_{Avg}$ | 3,2,3,3 | 2,6,7,11 | 1.678 |
| $Model_{W\times D}$ | $\theta_{HalfMax}$ | 3,2,3,3 | 2,6,7,11 | 1.678 |

Table 3.2: Intermediate Entity Selection.

tion process. As such, more "long" logical links exist between the clusters. In addition, it can be observed that the value of $Deviation$ is higher in the case of the Abilene network, which shows that the total cost in the sub-rings is on average smaller than the cost between the different sub-rings.

**Time Complexity Analysis** The time complexity of the proposed construction algorithms is influenced by the number of nodes in the management substrate, i.e. by the number of network edge nodes. It is necessary for each algorithm to determine the distance between each pair of $MS$ nodes, which is equivalent to obtaining the shortest paths between all pairs of nodes. This can be achieved with the Floyd-Warshall's algorithm [97] that can be implemented with a running time of $O(N^3)$, where $N$ is the number of nodes in the substrate. The time complexity of the ring construction algorithm is $O(N^2)$ [98]. In the case of the multiple rings construction algorithm, the time complexity depends both on the number of nodes in the substrate and the number of sub-rings that are formed. The latter is driven by the factor $\theta$, i.e. by the geographic distribution of edge nodes in the network. For each $MS$ node, the algorithm executes two main methods: it determines the closest neighbour of the node in the list of the $MS$ nodes that have not been considered yet and it applies the ring construction algorithm. Given that the minimum number of nodes allowed in each sub-ring is equal to 2, the maximum number of sub-rings that can be obtained is equal to $\frac{N}{2}$. In that case, the time complexity of the ring construction algorithm executed at each iteration is constant and equal $O(2)$, and the time complexity of the multiple rings construction algorithm is dominated by the size of the list of nodes to compare against at each iteration, i.e. $\frac{N(N+1)}{2}$ which gives a time complexity of $O(N^2)$. On the contrary, in the case where one sub-ring only is formed, the running time of the algorithm is mainly driven

by the complexity the ring construction algorithm, which is affected by the size of the sub-ring at each iteration. This gives a time complexity of $O(N^3)$. The average time complexity can be obtained by considering that there is an equal number of nodes in each sub-ring. In that case, the time complexity of the algorithm can be defined as $O(\frac{N^3}{r^2} + N^2)$ with $r$ the number of sub-rings. Finally, the intermediate entity selection algorithm consists in determining and comparing, for all nodes in each sub-ring, the distance to every other node in the substrate, and as such has an average complexity of $O(r^2N^2)$. In the case where all clusters consist of two nodes only, the complexity of the algorithm becomes $O(N^2)$.

The network environment considered in this work consists of an intra-domain fixed backbone network for which the maximum number of edge nodes are usually in the order of hundreds [104][105]. In addition, the construction of the management substrate is an offline process that is executed during the setup phase of the network. As such, the proposed algorithms can provide satisfactory performance in terms of time complexity and can be applied in practice by network operators.

### 3.7.3 Communication Cost Analysis

This section investigates how the number of nodes in the substrate, as well as the geographical distribution of the nodes, can affect the performance of each topology structure in terms of communication cost (delay). The ring structure considered in this section is obtained using the ring construction algorithm described in section 3.5.2, and the hybrid structure is obtained with the following parameters: $\theta_{\text{Avg}}$, $Model_{1\times \text{D}}$ and highest $C_{\text{LL}}$ as the initial node selection criterion, as defined in section 3.7.2.

**Round Trip Time Measurements** In order to evaluate the delay, the value of the round trip time (RTT) between pairs of nodes in the Abilene and the GEANT networks have been measured using the looking glass service available for these networks [106][107]. This provides an interface that allows the user to execute a ping command between routers in the network. This method was used in this work to record the values of the RTT between different pairs of nodes in both the Abilene and the GEANT networks. The evolution of the values of the RTT according to the geographical distance between the nodes is shown in figure (3.10). As observed, the value of the RTT linearly increases with the distance, which is consistent with previous studies [95][96].

**Comparison of the Management Substrate Communication Costs** Based on the values of the RTT, experiments are performed to compare the communication cost in terms of the maximum delay incurred in each topology structure. The communication cost is defined according

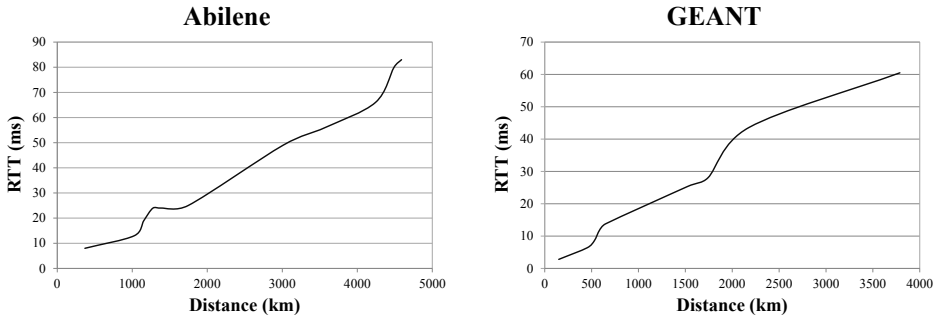



Figure 3.10: Evolution of the RTT as a function of the distance in case of the Abilene and GEANT networks.

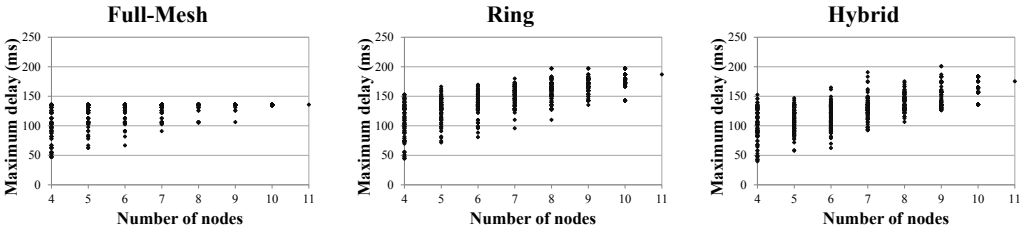



Figure 3.11: Evolution of the maximum delay according to the number of management substrate nodes in case of the Abilene network.

to the communication model used in each structure as follows. In the case of the full-mesh model, the communication cost is given by the maximum delay between any pair of nodes in the substrate. In the case of the ring model, this corresponds to the total time required for a message to travel around the ring. In the hybrid model, the communication cost is defined as the sum of the maximum delay between $IE$ nodes and the maximum sub-ring cost. As such, the values considered in each model represent an upper bound in terms of delay for each of the structures.

The experiments consist of a set of substrate configurations defined for each network topology (i.e. Abilene and GEANT), so that for each configuration, a number of nodes ($N$) is randomly selected and connected according to the proposed structures. $N$ varies from $4$ to the total number of nodes in the network. In the case of 3 nodes and less, the different models lead to identical structures. In order to represent various geographical distributions of nodes in the substrate, a large number of configurations (i.e. choice of network nodes) is considered for each value of $N$. The results of the experiments are shown in figures (3.11) and (3.12), where the communication cost (maximum delay) obtained in each topology structure for different number of substrate nodes is presented for the Abilene and GEANT networks, respectively.

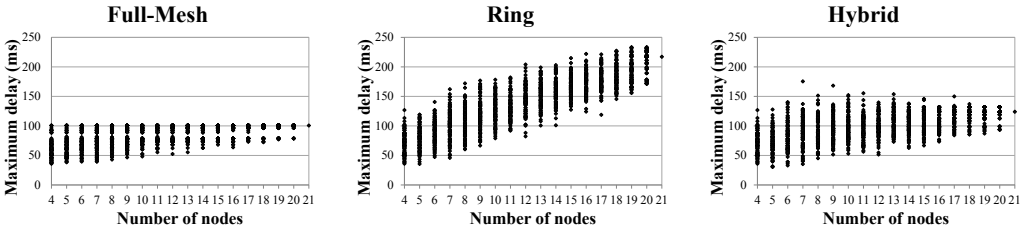



Figure 3.12: Evolution of the maximum delay according to the number of management substrate nodes in case of the GEANT network.

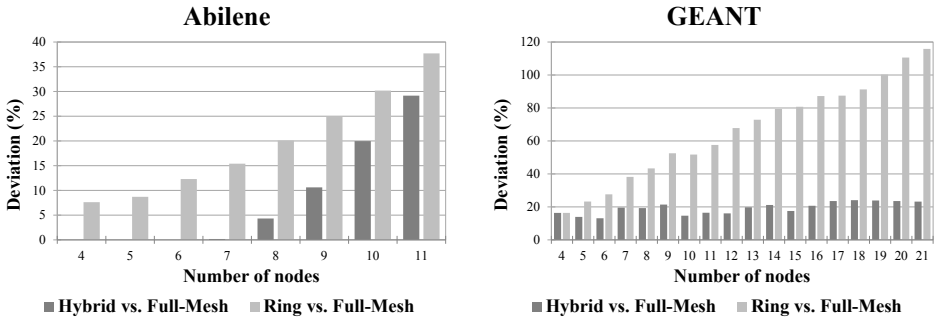



Figure 3.13: Evolution of the average deviation in terms of delay between the hybrid and full-mesh models, and, between the ring and full-mesh models, in case of the Abilene and GEANT networks.

As can be observed, the communication cost obtained for a given number of nodes $N$ form, in all the cases, a vertical line parallel to the y-axis. This confirms that the maximum delay is influenced by the geographical distribution of the nodes. It can be noted, however, that in the case of the full-mesh model, there exists an upper bound in terms of the maximum value that the delay can reach, which is independent of the number of nodes in both the Abilene and GEANT networks. As a result, it can be deduced that the communication cost is mainly driven by the maximum distance between $MS$ nodes in the case of the full-mesh model. In the case of the ring model, it can be observed that the maximum delay does not only depend on the geographical distribution of nodes but is also driven by the number of nodes in the substrate. The value of the communication cost increases as the number of substrate nodes increases. While the cost in the hybrid model is also affected by the number of nodes, the influence of this factor is lower compared to the ring model.

**Average Deviation Analysis** The deviation between the communication cost obtained with (i) the hybrid model and the full-mesh model, and (ii) the ring model and the full-mesh model is finally investigated. Figure (3.13) represents the average deviation in percentage obtained in each case for the Abilene and the GEANT networks.

As observed, the average deviation is always positive[1], which shows that the full-mesh model performs better than the two other approaches in terms of communication cost. In addition, it can be noted that the performance obtained by the three models is similar when the number of nodes is small, especially in the case of the Abilene network where the average deviation is below $10\%$. As the number of nodes increases, however, the full-mesh model outperforms the two other models. The performance of the ring model, in particular, becomes much worse than that of the full-mesh when the number of substrate nodes increases. A deviation of more than $100\%$ is obtained in the case of the GEANT network with $21$ $MS$ nodes. In contrast, the difference in terms of performance between the hybrid model and the full-mesh model is much less significant. A maximum of $30\%$ deviation can be observed in the case of Abilene and around $25\%$ in the case of GEANT. It is finally interesting to note that, in the case of GEANT, the average deviation between the hybrid and the full-mesh model is not significantly affected by the number of nodes in the substrate.

### 3.7.4 Management Substrate Structure Comparison Summary

As shown in the previous section, the performance of the ring model, in terms of communication cost, is very sensitive to the number of nodes in the substrate, i.e. to the number of decision-making points, and as such, this model presents some scalability limitations. In contrast, the performance of the full-mesh and the hybrid models are mainly affected by the geographical coverage of the network. In order to efficiently support the adaptive resource management scheme, the delay incurred by the communication between the decision-making points needs to be kept small. Therefore, the choice of a ring structure is not recommended when the number of substrate nodes exceeds $5$ or $6$.

In addition to the performance in terms of delay, the topological characteristics of the structures can also influence the choice of the model to apply. Due to its high degree of connectivity, the full-mesh topology model provides a greater flexibility to select which nodes to communicate with. The full-mesh structure, however, requires that every node in the substrate maintains locally information about every other $MS$ node, which may also raise some scalability limitations. The hybrid model offers a trade-off between the full-mesh and the ring models. It outperforms the ring model in terms of communication cost and can also offer competitive performance compared to the full-mesh approach. Furthermore, it can overcome the limitations of the full-mesh model by restricting the volume of substrate information that needs to be maintained at each $MS$ node.

[1] In the case of Abilene, the average deviation obtained between the hybrid and full-mesh models is close to zero when the number of nodes is smaller than 7 and as such, these results are not visible in the figure.

| Model | Full-mesh | Ring | Hybrid |
|---|---|---|---|
| **Level of hierarchy** | Flat structure | Flat structure | One level hierarchy |
| **Number of neighbours** | N-1 | 2 | min $= 2$<br>max $= 2 + (r - 1)$ |
| **Communication model** | Star fashion mechanism | Hop-by-hop mechanism | Star & hop-by-hop mechanisms |
| **Communication cost** | Driven by the geographical distance | Driven by the number of nodes | Driven by both factors |

Table 3.3: Comparison of the Proposed Management Substrate Topology Models.

The characteristics of the three models investigated in this chapter are summarised in Table (3.3). The total number of nodes in the substrate is noted $N$ and the total number of sub-rings in the hybrid model is noted $r$.

## 3.8 Conclusion

In this chapter, a new adaptive in-network management framework was presented. The proposed framework relies on a decentralised decision-making process distributed over the network edge nodes. These decide in a coordinated fashion which configuration changes to apply, according to current conditions in the network, so that network resources can be better utilised. In order to support this decentralised and coordinated decision-making process, the concept of a management substrate to organise the participating nodes into the reconfiguration process is introduced. The management substrate is defined as a logical infrastructure that facilitates the exchange of information between decision-making points. Three different topology structures to organise the nodes in the management substrate are investigated. The characteristics of each model are described and compared and a set of offline algorithms that can be used in practice to construct the proposed structures are designed. The choice of the different design parameters are discussed and evaluated. Finally, a thorough quantitative and qualitative evaluation of the impact of key parameters (i.e. the number of nodes in the substrate and their geographical distribution) on the performance of the different models in terms of communication overhead was performed. The analysis of the results of the evaluation provides useful indications about the use of a particular structure for specific network settings and characteristics. The results show that the use of a ring model to connect the substrate nodes can lead to poor performance in terms of communication cost. As such, this model is not considered in the subsequent chapters of this thesis. The work presented here is extended in the following chapters in which the proposed in-network management framework is applied to specific resource management scenarios.

## Chapter 4

# Load Balancing-aware Adaptive Resource Management

## 4.1 Introduction

To cope with unexpected traffic variations and network dynamics, traffic engineering approaches that can adapt to network and traffic dynamics are required. In contrast to offline schemes, online traffic engineering approaches do not rely on the knowledge of a traffic matrix to configure the routing or the link weights. Instead, they dynamically adapt the settings in short timescales based on real-time information from the network in order to rapidly respond to traffic dynamics [65]. Despite recent proposals to enable adaptive traffic engineering in ordinary IP networks [108][71][74], current approaches usually rely on a centralised traffic engineering manager to periodically compute new configurations according to dynamic traffic behaviors.

This chapter presents a novel intra-domain resource management approach for IP networks, in which the traffic distribution is controlled in an adaptive and decentralised manner according to the network conditions. Based on path diversity provided by multi-topology routing, the traffic between any source-destination (S-D) pair is balanced across several paths according to splitting ratios, which are (re)-computed by the network edge nodes (i.e. source nodes) themselves [6][7]. New configurations are not computed by a centralised management entity, but instead, are the result of a real-time adaptation process executed directly by the source nodes. To decide upon the most appropriate set of reconfiguration actions to perform, nodes coordinate among themselves through the management substrate infrastructure described in chapter 3. This chapter describes the details of the adaptation process executed by the substrate nodes for resource optimisation purposes and elaborates on the specific algorithm for periodical reconfigurations. It also investigates how network edge nodes coordinate through the management substrate structure and presents the interaction models designed in the case of the full-mesh and the hybrid substrate structures. The performance of the proposed scheme is evaluated based on

realistic topologies and real traffic traces. Results show that near-optimal performance can be achieved in terms of resource utilisation in a scalable and responsive manner.

The rest of the chapter is organised as follows. Section 4.2 discusses related work on online traffic engineering and multi-topology routing. Section 4.3 presents the main features of the proposed scheme and describes an algorithm to compute the different routing planes. Section 4.4 details the principles of the adaptation process and the reconfiguration algorithm. In section 4.5, different models of interaction between substrate nodes to support the adaptation process are investigated. Section 4.6 presents the results of the overall performance evaluation of the approach. Finally, the main contributions of this work are summarised in section 4.7.

## 4.2 Related Work

### 4.2.1 Online Traffic Engineering

Current practices for managing resources in fixed networks rely on offline approaches, where a centralised management system is responsible for computing routing configurations that optimise the network performance over long timescales, e.g. weekly or monthly. Given their static nature, these approaches can be sub-optimal in the face of unexpected traffic demand. To cope with their limitations, new traffic engineering schemes that can adapt to network and traffic dynamics are required. In order to rapidly respond to traffic variations, online traffic engineering approaches dynamically adapt the settings in short timescales according to real-time information received from the network [65].

The earliest examples of routing schemes that can adapt to network conditions were developed in the context of the ARPANET [109][110]. The main objective was to define link weights as a function of the traffic load in order to adapt the routing to network conditions. However, these approaches had several limitations especially due to the network instability incurred by the re-computation of link weights. More recently, some research efforts have exploited the features of multi-path routing to adapt the distribution of traffic load in the network according to changing conditions. The volume of traffic (represented by splitting ratio) sent across several available paths between each source-destination pair of nodes in the network is dynamically adjusted based on real-time traffic information in order to optimise a given objective function. More specifically, the objective is to distribute the traffic load as evenly as possible onto the different parts of the network in order to accommodate unanticipated demand or traffic variations. There have been some proposals for both online MPLS-based traffic engineering such as [72][73] and online IP-based traffic engineering such as [108][71][74].

In MPLS-based approaches, splitting decisions are enforced at ingress routers [72][73]. In

[72], the load balancing decisions are performed through a centralised management entity that has a global knowledge about the network state and the configuration of ingress routers. Unlike [72], the approach presented in [73] does not rely on any centralised system to perform the reconfigurations. It applies some of the principles of the TCP congestion control mechanism to traffic engineering. More precisely, each ingress-egress pair of nodes is represented by a pair of agents implemented in the relevant ingress and egress routers. The egress node agent uses load information received from the different routers on the path from the ingress router to compute some feedback. This is then provided to the ingress node agent to compute the new splitting ratios, so that different adjustment decisions can be made independently by each pair of agents without any explicit coordination. In order to guarantee consistencies between the different adjustment decisions, a control mechanism which involves global communication between nodes in the network needs, however, to be deployed.

Compared to MPLS-based approaches, the decision-making process developed in [108] is distributed over all the network nodes. Each node can dynamically adjust the splitting ratios between the different available next hops based on feedback information received from upstream routers. The main principle of [108] is inspired from [111], an early proposal to extend the Open Shortest Path First (OSPF) [112] routing protocol with adaptive traffic engineering (i.e. load balancing) capabilities. In [111], each network node uses network-wide load information distributed through a flooding mechanism to adjust the volume of traffic that is forwarded towards the different next hops, and, as such, to balance the traffic load between the different parts of the network. In order to overcome the limitations of [111] in terms of scalability, a new method to disseminate load information was proposed in [108]. Unlike [111] where each node has the same global knowledge about the network state, information received from direct neighbours only is used in [108] to make load balancing decisions. However, the consistency between independent decisions taken by each network node individually cannot be guaranteed. A similar signalling mechanism was considered in [71]. Compared to [108] where load information is not differentiated per-destination, the dissemination method used in [71] allows each network node to compute and report load statistics on a per destination basis. As such, each decision-making agent can monitor the effect of splitting ratio adjustments for each destination, which allows finer control in terms of traffic distribution. In order to guarantee consistency between independent decisions, the load balancing algorithm is designed according to game theory principles. Although better performance in terms of scalability can be obtained with such a hop-by-hop information dissemination mechanism, the signalling communication overhead can still be an issue in practice given that all nodes need to communicate to exchange information about the

current state of the network. In addition, specific structures need to be implemented in all network nodes to maintain and process the relevant traffic engineering information and decisions.

To avoid flooding the network with signalling messages, a scheme by which nodes use only local information from their direct outgoing links to make routing decisions was proposed in [113]. Due to the local scope of information regarding network conditions, this approach does not target optimality but robustness. It has been argued, however, that approaches that focus on robustness often have poor performance in terms of resource utilisation in lightly loaded network conditions [114]. Furthermore, since decisions are based on a local view of the network state only, the consistency between multiple concurrent decisions cannot be guaranteed.

In contrast to the proposed ratio adjustment approaches, recent work in [115] has investigated a new method to dynamically adapt link weights in reaction to load variation. In order to overcome the limitations of previous results in terms of instability, the weights are sequentially modified in one link at a time. However, it was recently argued that changing link weights too often should be avoided since this may lead to instability and service degradation during the network state re-convergence process [116].

### 4.2.2 Multi-Topology Routing

There has been over the years an increasing interest for the use of virtual planes to logically represent a physical network infrastructure according to different configurations. Multi-Topology Routing (MTR) [117][118] is a standardised extension to the common Interior Gateway routing Protocols (IGP), i.e. Open Shortest Path First (OSPF) and Intermediate System to Intermediate System (IS-IS), that aims to define from a physical network topology several independent virtual IP topologies, each having its own routing configurations (i.e. link weight settings). Based on the link weight settings, the Shortest Path First (SPF) algorithm can be applied between each source-destination pair of nodes in each topology independently. It is therefore possible to compute a set of paths between each pair of end points in the network, each path being associated with one of the virtual topologies. In order to determine to which topology packets need to be routed, these are marked at each source node with the Multi-Topology Identifier (MT-ID) of the routing topology to which the corresponding traffic flows have been assigned. A separate routing table needs to be implemented for each topology (i.e. routing scheme) in each router, so that upon receiving a traffic flow, the router analyses the MT-ID marked in the packets and forwards the packets to the next-hop according to the relevant routing table [119].

While multi-topology extensions were initially developed for the purpose of routing different types of services and traffic across different paths in the network, these have been more recently considered for other purposes such as network resilience [120][121][119] and intra-

domain traffic engineering [122][123][74].

Compared to MPLS solutions, in which the traffic load can be split arbitrarily over multiple available paths, traffic engineering functionality used in ordinary IP network environments mainly relies on the equal cost multi-path (ECMP) [2] feature. This requires the equal distribution of traffic load over multiple next hops of the equal cost paths between any source-destination pair of nodes. The performance that can be achieved in terms of load balancing in ordinary IP environment is therefore limited. It was argued in [122] that MTR features could be used to overcome the limitations of current practices. More specifically, this showed that by exploiting the path diversity provided through the configuration of different virtual planes, arbitrary traffic splitting and flexible load balancing can be enforced, which, as a result, allows a finer control of the traffic load distribution in the network. Based on the configuration of $n$ virtual topologies, the traffic demand between any source-destination pair of nodes is logically partitioned into $n$ independent sets at the source nodes. Each traffic set is then assigned to one of the $n$ virtual topologies and routed according to the topology's configuration. The volume of traffic in each virtual topology is driven by splitting ratios that are precomputed according to the traffic demand in order to optimise the utilisation of network resources. The use of MTR for traffic engineering purposes was further advocated in [123], where a method to compute the logical topologies is proposed.

An adaptive traffic engineering approach that relies on MTR infrastructure is presented in [74]. The proposed framework consists of two components: an offline link weights computation algorithm, to configure the different topologies, and an online centralised adaptation algorithm, to dynamically adjust the traffic splitting ratios of the different routing planes. However, using a centralised solution to perform adaptive resource management may raise some issues. In addition to single point of failure limitations, there may be some issues in terms of scalability incurred by the communication overhead between the central manager and devices at runtime, and lag in the central manager's reactions that may result in sub-optimal performance.

A challenging question when implementing a solution that relies on MTR is the actual number of virtual planes that are required to achieve load balancing objectives. With MTR, the size of the routing table that needs be computed in each router is proportional to the number of topologies. This may have some implementation implications especially in terms of storage cost. It was shown in [122][123][74] that only a small number of topologies (typically between 3 and 5) is enough in practice to offer pretty good path diversity and that 2 or 4 traffic sets are sufficient to achieve near-optimal performance [122].

## 4.3 Load Balancing Management Application

### 4.3.1 Notations and Definitions

This subsection introduces the main notations used in this chapter. In a similar manner to the notations used in chapter 3, let $\mathcal{L}$ be the set of network links and $\mathcal{N}$ the set of network nodes. Let $\mathcal{N}_{\mathrm{E}}$ represent the set of network edge nodes, i.e. network nodes generating and absorbing traffic, and $\mathcal{N}_{\mathrm{C}}$ the set of network core nodes. Let $c(l)$ represent the capacity of link $l \in \mathcal{L}$. The utilisation of any link $l$ in the network, noted $u(l)$, is defined as the ratio of the traffic load $\rho(l)$ over $l$ and the capacity $c(l)$, i.e. $u(l) = \frac{\rho(l)}{c(l)}$.

Let $u_{\mathrm{max}}$ be the maximum utilisation in the network defined so that:

$$u_{\mathrm{max}} = \max_{l \in L}(u(l)) \tag{4.1}$$

Let $l_{\mathrm{max}}$ represent the link with the maximum utilisation in the network. For any pair of edge nodes $(i, j) \in \mathcal{N}_{\mathrm{E}}^2$, let $sd_{\mathrm{ij}}$ represent the source-destination pair of source node $i$ and destination node $j$. A source-destination pair $sd_{\mathrm{ij}}$ is associated with a volume of traffic $V(sd_{\mathrm{ij}})$ that represents the traffic demand between source node $i$ and destination node $j$. The traffic flow $F(sd_{\mathrm{ij}})$ of source-destination pair $sd_{\mathrm{ij}}$ is defined as the couple $(sd_{\mathrm{ij}}, V(sd_{\mathrm{ij}}))$. Let $\phi_{\mathrm{i}}$ be the set of traffic flows locally originating at edge node $i$ so that:

$$\phi_{\mathrm{i}} = \left\{ F(sd_{\mathrm{ij}}), \quad j \in \mathcal{N}_{\mathrm{E}} \right\} \tag{4.2}$$

### 4.3.2 Approach Overview

In this chapter, a new network resource management scheme, by which the distribution of the traffic load in the network can be controlled in an adaptive and decentralised manner, is presented. Instead of relying on decisions received from a centralised manager, network edge nodes coordinate among themselves to decide how to re-distribute the traffic load onto the different parts of the network in order to better utilise the network resources. More specifically, based on the path diversity provided by configuring several virtual topologies, the traffic between any source-destination pair of nodes is balanced across several paths according to splitting ratios which are used to distribute incoming packets at source nodes. Splitting ratios are not predetermined by an offline process as in other approaches, e.g. [122], but are instead dynamically (re-)computed by the source nodes (i.e. network edge nodes) themselves according to run-time information about the network state, without centralised control (as is the case in [74]). To provide a set of possible routes between any of the source-destination pairs, the proposed scheme relies on multi-topology routing. The distribution of the traffic load is controlled through reconfiguration actions executed at the source nodes that dynamically adjust the proportion of traffic

assigned to each topology, so that the traffic is periodically re-balanced from the most utilised links towards less loaded parts in the network. Splitting ratios are decided by source nodes only and are not modified by other nodes on the route (i.e. no further splitting is permitted along the path).

The objective of this adaptive control is to minimise the utilisation of the most utilised link in the network. Minimising the maximum utilisation in the network is a commonly used traffic engineering objective in the literature (e.g. [68][69][70][73]). This enables the reduction of hotspot utilisation against dynamic traffic behaviours, and, as such, the reduction of congestion risks by spreading the load over available paths [64]. Given that the objective is to distribute the traffic load as evenly as possible in the network, this also increases the likelihood that future traffic demand can be satisfied.

Performing edge-based control of traffic provides several advantages. It offers better scalability since network edge nodes only are involved in the reconfiguration process. In addition, given that edge nodes have detailed information about the incoming traffic flows, they can perform a fine-grained control on how packets are distributed in the different topologies [122]. Packets that belong to the same TCP flow are always assigned to the same topology and no further adjustment is permitted along the route. This ensures that all packets in one flow follow only a single path to the destination, which avoids packet out-of-delivery issues that deteriorate the performance of TCP [124].

The adaptation process is performed in short timescales, for instance, in the order of 5-15 minutes, which is in accordance with the common network monitoring interval [93][74]. It should be noted that these assumptions for the network environment call for mechanisms with a certain level of network environment awareness. Most of the proposals use integrated monitoring facilities (e.g. [71][74]) that may be restricted to the studied context or not detailed at a sufficient level. It can be argued that the monitoring or information management system could be decoupled from the resource management solution. It is a complicated aspect that should be investigated independently. Example platforms that could be used are [83][125][84]. In this thesis, it is assumed that the information management overlay proposed by Mamatas et al. in [83] can be used to provide the necessary network information. More specifically, the proposed framework is a management infrastructure that performs monitoring functions such as collecting, processing and disseminating information from/to the network entities and management applications. It consists of a controller entity that performs overlay-wide control of the platform and enforces decisions by communicating with management nodes distributed over different points in the network. The number and placement of management nodes is decided in

an offline fashion and depends on the management application. In this chapter, it is assumed that the monitoring information is provided at the beginning of the reconfiguration process to all network edge nodes though the management nodes forming the information overlay.

### 4.3.3 Virtual Topology Configuration

The configuration of the different virtual planes is part of an offline process that takes as input the underlying physical network infrastructure. The virtual topologies need to satisfy two objectives:

1. To provide a set of non-completely overlapping paths between any source-destination pair of nodes in the network, i.e. there is always at least one path which is not completely overlapping with the others.

2. To avoid introducing critical links, i.e. given a link $l$ that is traversed by some traffic from node $i$ to node $j$, there always exits an alternative path that can be used for routing the traffic without traversing $l$.

The idea of obtaining topologies that satisfy these requirements is the following. Assume that $l$ gets congested. It should be possible to move some traffic away from this link towards other parts of the network, i.e. towards other links. By computing topologies which satisfy the above requirements, it is ensured that for any link $l$ in the network, it is always possible to find at least one traffic flow that is routed over link $l$ in a set of topologies while it does not traverse $l$ in the set of the other topologies.

Figure (4.1) illustrates a simple example of how virtual topologies that satisfy the aforementioned requirements can be derived from a base physical topology. In this example, traffic between the source-destination pair $1-3$ is forwarded from source node $1$ towards destination node $3$. In each of the alternative topologies $T_1$, $T_2$ and $T_3$, some links are assigned a *MAXIMUM* weight (represented here with infinity) which prevents these links from being used for routing the traffic demand between node $1$ and node $3$. With these settings, three non-overlapping paths can be determined between node $1$ and node $3$: (1;4;2;3), (1;4;5;3) and (1;2;3) and no critical link is created. The configuration of the alternative topologies is represented at the network level by associating a vector of link weights to each link in the network, each component of the vector being related to one topology. In this simple example only three virtual topologies are required to satisfy the objectives.

To obtain the desired virtual planes in more complex and realistic network environments, where each source-destination pair has to be taken into account, is not straightforward. The next paragraph presents an offline algorithm that relies on the characteristics of the physical network

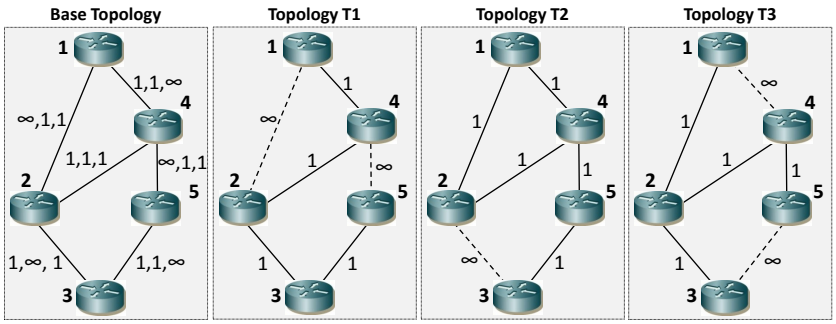



Figure 4.1: Building multiple topologies.

topology to compute a set of virtual planes that can satisfy objectives 1 and 2. It should be noted that balancing the traffic over the different paths provided by multi-topology routing may lead to route traffic over paths with longer round trip times. Best effort traffic is considered in this work and as such, this is not an issue.

**Virtual Topologies Computation Algorithm** The proposed algorithm to compute the virtual topologies follows the method described in [123]. Initially, $n$ copies $T_{\mathrm{k},\, k\in[1;n]}$ of the original physical topology $T_{\mathrm{base}}$ are created. A subset of links are then removed from each topology $T_{\mathrm{k}}$ in a round-robin manner, so that each topology still remains connected, i.e. it is possible to reach all nodes. The pseudo-code of the algorithm is presented in Algorithm (4.3.1).

Links are initially ranked in decreasing order of the inverse of their capacity, i.e. links with the smallest capacity are first in the list. These are then considered iteratively by the algorithm that determines from which topology each link can be removed. A link can be removed if this does not lead to the disconnection of the topology. There are by definition links that cannot be removed without causing a disconnection. Examples of such links are links associated with a node with a degree of connectivity equal to one. Such links are disregarded by the algorithm at the beginning of the process. In order to determine from which topology link $l$ can be removed, the algorithm considers each plane $T_{\mathrm{k}}$ sequentially until it can find one that satisfies the requirements. In order to balance the number of links removed in each topology, the order by which the topologies are considered is modified for each new link to test. The reasons for considering links in the decreasing order of the inverse of their capacity are the following. Given that links with the smallest capacities are more likely to become highly loaded, it is expected that traffic will need to be diverted away from these links. As such it is important that there exist alternative topologies in which they are not used. In the case where the number of topologies to use does not permit all links to be removed, this ensures that the path diversity requirements can be satisfied for the most sensitive links.

**Algorithm 4.3.1** Pseudo-code to compute the logical topologies.

**Notations**

$L$ set of links in the network

$n$ number of desired virtual topologies

**Pseudo-code**

Create $n$ copies $T_{\mathrm{k},\, k\in[1;n]}$ of the physical topology

Sort links in $L$ by decreasing order of the inverse of their capacity

$l \leftarrow$ first link in the ordered list

$k \leftarrow 0$

**while** allLinksAnalysed = false **do**

findTopo = false

nbCheckTopo = 0

**while** findTopo = false **and** nbCheckTopo $\leq n$ **do**

Try to remove $l$ from $T_{\mathrm{k}}$

nbCheckTopo++

**if** $T_{\mathrm{k}}$ still connected **then**

Remove $l$ from $T_{\mathrm{k}}$

findTopo = true

**else**

$k \leftarrow k \mod n+1$

**end if**

**end while**

**if** next link in the ordered list is non null **then**

$l \leftarrow$ next link in the ordered list

$k \leftarrow k \mod n+1$

**else**

allLinksAnalysed = true

**end if**

**end while**

The computed topologies are then used to build the vector of link weights assigned to each link in the physical network. Links that are removed from a topology are represented by the *MAXIMUM* value in the vector of link weights while the weight of the other links is set to the original value in the physical base topology. It was conjectured by experiments in [123] that fewer than five logical topologies are typically needed to cover all links with this method. The results presented in section 4.6.2 show that four topologies are enough for the networks considered in this chapter.

**Time Complexity** The algorithm considers iteratively each network link to determine from which virtual topology it can be removed. For each topology, it checks whether the connectivity can be maintained when the link is removed. The algorithm to check the connectivity of a topology has a time complexity of $O(L)$ [97], where $L$ is the total number of links in the graph. As a result, the running time of the virtual topologies computation algorithm is equal to $O(n \cdot L^2)$, where $n$ is the total number of virtual topologies to consider. Given the characteristics of the network environment considered in this work (i.e. intra-domain backbone networks [104][105]), it can be concluded that the proposed algorithm can be used in real deployment to compute the virtual topologies in an offline fashion.

### 4.3.4 Management Substrate of Coordinated Entities

For load balancing application, the network edge nodes are organised into a management substrate (MS) as described in chapter 3. The MS is a logical structure used to facilitate the exchange of information between decision-making entities for resource management purposes. The substrate is built in an offline fashion during the initial configuration of the network according to the methods presented in chapter 3. Its formation is based on the identification of the edge nodes in the physical network. In the case of a PoP (Point of Presence) level network, for instance, each node is a potential source of traffic and is therefore added to the substrate. In this work, the use of the full-mesh and the hybrid topology structures were considered to connect the nodes in the substrate. The details of the implementation of the different structures are presented in section 4.5. Nodes in the MS coordinate between themselves through an adaptation process, which consists of a sequence of reconfiguration actions to compute new splitting ratios.

To perform a reconfiguration involves adjusting the traffic splitting ratios for some of the source-destination pairs for which traffic is routed across the link with the maximum utilisation in the network, $l_{\text{max}}$. This means that more traffic is assigned to topologies not using $l_{\text{max}}$ to route traffic thus decreasing the traffic volume assigned to topologies that do use $l_{\text{max}}$.

In realistic scenarios, links in the network are traversed by multiple traffic flows (i.e. they

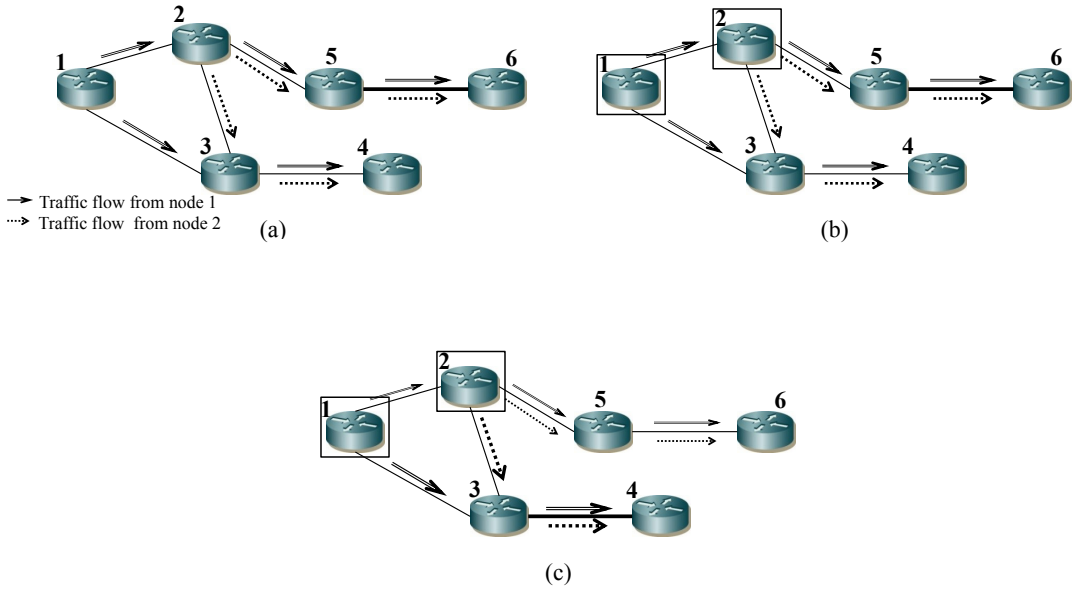



Figure 4.2: Example of conflicting decisions between node $N_1$ and node $N_2$.

are used by several source-destination pairs to route the corresponding traffic demand) and therefore, several MS nodes may be eligible to adapt the ratios of traffic flows traversing $l_{\max}$. Due to the limited network view of individual substrate nodes, actions taken by more than one node at a time may lead to inconsistent decisions, which may jeopardise the stability and the convergence of the overall network behaviour [74]. For instance, in the process of shifting traffic away from $l_{\max}$, the different reacting nodes can re-direct traffic flows towards the same links, as depicted in figure (4.2), thus potentially causing congestion. In figure (4.2), source nodes $N_1$ and $N_2$ send traffic over link $l_{5-6}$. In (a), $l_{5-6}$ is identified as being the most utilised link in the network. $N_1$ and $N_2$ both react to this information (b) and decide to perform some reconfigurations locally, which results in more traffic from both $N_1$ and $N_2$ routed towards link $l_{3-4}$, which then becomes overloaded (c).

To avoid such inconsistent decisions, the reconfiguration actions are decided in a coordinated fashion between nodes in the substrate. At each sequence/iteration of the adaptation process, one node in the substrate - called the Deciding Entity (DE) - is selected to initiate reconfiguration actions. The selected DE is responsible for executing the reconfiguration algorithm over its locally originating traffic flows with the objective to re-balance the network load. The DE role can be taken by any substrate node. In order to select a unique DE, a selection logic is therefore implemented in the different MS nodes, so that each node can determine independently whether or not it can assume the DE role for the new reconfiguration interval. The selection mechanism depends on the substrate structure. The details of the process are presented in section 4.5.

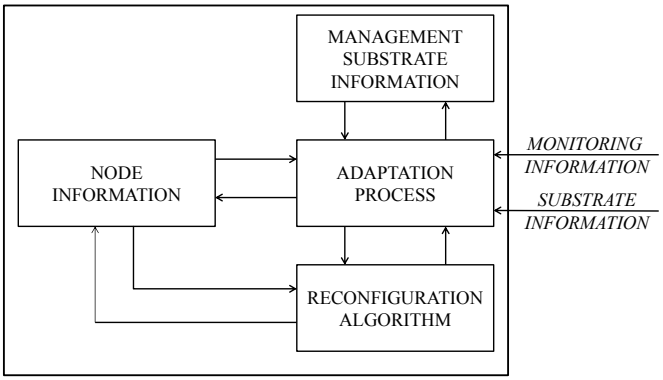



Figure 4.3: Components overview at the edge node level.

### 4.3.5 System Design

In order to realise the proposed adaptive resource management scheme, a set of components need to be deployed at each management substrate (MS) node as depicted in figure (4.3). The *Adaptation Process* component manages the coordination of the node with other MS nodes during the reconfiguration process. It controls the execution of a *Reconfiguration Algorithm*, which consists of the set of methods to adjust the splitting ratios of the traffic flows originating at that node. In addition, information is locally maintained. Information about the locally originating traffic flows is stored in the *Node Information* component and information about the other MS nodes is maintained in the *Management Substrate Information* component.

Information in the *Node Information* component is stored in two tables referred to as the Link Information Table (LIT) and the Demand Information Table (DIT) respectively - Tables (4.1) and (4.2) show the structure of each table at edge node $i$. The LIT contains information about the links used in the paths of all the locally originating flows. For each link, it stores the link capacity, references to the source-destination pairs $sd$ that use that link for routing their associated traffic flow in at least one topology, and for each of these $sd$ pairs, the involved topology(ies). Based on this information, each edge node can efficiently determine if a local flow contributes to the load of a link in the network, and in that case, on which topology(ies). It also maintains for each $sd$ pair a parameter, noted $m_{\text{sd}}(l)$, that is used to evaluate whether the traffic flow should be considered for traffic removal. Details about this parameter are provided in section 4.5. The DIT contains dynamic information related to each flow entering the network at the source node. It maintains the current and updated splitting ratios assigned to each topology, noted $x_{\text{T}}^{\text{cur}}(sd)$ and $x_{\text{T}}^{\text{upd}}(sd)$ respectively, as well as the associated traffic volume $v_{\text{sd}}$ for the current time interval. It also maintains a binary variable $\sigma_{\text{sd}}$ that indicates whether the flow has been modified during the current reconfiguration interval.

| | | | | |
|---|---|---|---|---|
| $l$ | $c(l)$ | $sd_{\text{i1}}$ | $m_{\text{i1}}(l)$ | $T_1$ |
| | | | | ... |
| | | | | $T_{\text{n}}$ |
| | | ... | ... | ... |
| | | $sd_{\text{iN}}$ | $m_{\text{iN}}(l)$ | $T_1$ |
| | | | | ... |
| | | | | $T_{\text{n}}$ |

Table 4.1: Entry of the Link Information Table (LIT).

| | | | | | |
|---|---|---|---|---|---|
| $sd_{\text{ij}}$ | $v_{\text{sd}_{\text{ij}}}$ | $\sigma_{\text{sd}_{\text{ij}}}$ | $T_1$ | $x^{\text{upd}}_{\text{T}_1}(sd_{\text{ij}})$ | $x^{\text{cur}}_{\text{T}_1}(sd_{\text{ij}})$ |
| | | | ... | ... | ... |
| | | | $T_{\text{n}}$ | $x^{\text{upd}}_{\text{T}_{\text{n}}}(sd_{\text{ij}})$ | $x^{\text{cur}}_{\text{T}_{\text{n}}}(sd_{\text{ij}})$ |

Table 4.2: Entry of the Demand Information Table (DIT).

Based on the information stored in the LIT and DIT tables and received through the management substrate, each edge node can determine its current state, which can be either idle or active depending on whether or not it needs to perform a reconfiguration. When in the latter state, the node executes the reconfiguration algorithm over its locally originating flows to determine the new splitting ratios that can decrease the utilisation of the most utilised link in the network. If with these new splitting ratios no other link in the network gets overloaded, the adjusted splitting ratios are updated in the DIT and enforced at the next time interval.

## 4.4 Adaptive Resource Management Scheme

### 4.4.1 Adaptation Process

The overall objective of the proposed adaptive resource management scheme is to balance the load in the network by moving some traffic away from highly utilised links towards less utilised ones in order to reduce the utilisation of the hotspots against dynamic traffic behaviours.

To achieve the load balancing objective, the proposed scheme successively adjusts the splitting ratios of traffic flows through a sequence of reconfiguration actions that constitute the *adaptation process*. At each iteration, the splitting ratios of some of the traffic flows routed over the link $l_{\text{max}}$ with the maximum utilisation in the network are adjusted so that some traffic can be moved away from $l_{\text{max}}$. In order not to overload other parts in the network, the maximum volume of traffic that can be diverted is defined so that no link $l$ attains a utilisation higher than the original value of the utilisation of $l_{\text{max}}$, $u_{\text{max}}$, i.e.

$$\forall l \in \mathcal{L}, \quad u(l) \leq u_{\text{max}} \tag{4.3}$$

A situation that should be avoided is that excessive traffic is diverted to a link which is originally lightly utilised, so that its new utilisation becomes higher than $u_{\max}$. To prevent inconsistencies between concurrent traffic splitting adjustments, the different reconfiguration actions are taken iteratively, so that at each iteration, one node in the management substrate, called the Deciding Entity ($DE$), is selected to perform some reconfigurations over its locally originating traffic flows. The reconfiguration decisions are the result of a *reconfiguration algorithm*, which aims at adjusting the splitting ratios of as many local flows as possible so that the maximum utilisation in the network can be decreased. The flows for which adjustments are performed are marked as *modified* through the binary variable $\sigma_{\text{sd}}$ introduced in section 4.3.5, and the new ratios are copied in the updated ratio entries in the DIT.

The adaptation process terminates if a successful configuration cannot be determined or if it reaches the maximum number of permitted iterations (a parameter of the algorithm/system). The entries in the LIT and DIT tables are reset at each substrate node at the end of the process and the new ratios are enforced.

### 4.4.2 Reconfiguration Algorithm

**Reconfiguration Process** The *reconfiguration algorithm* is executed by the DE node at each iteration of the *adaptation process*. It follows a greedy process that successively recomputes the splitting ratios of the local traffic flows in $\phi_{DE}$. More specifically, the reconfiguration process works as follows. Based on the information received from other nodes in the substrate, the algorithm determines the link with the maximum utilisation, $l_{\max}$. It tries to adjust the splitting ratios of the flows in $\phi_{\text{DE}}$ which contribute to the load on $l_{\max}$ in order to move traffic away from the link. The outcome of the algorithm at each iteration is either positive, which means that part of a local flow can be diverted from $l_{\max}$, or negative if this is not possible.

The algorithm first identifies the flows in $\phi_{\text{DE}}$ that can be diverted from $l_{\max}$. A flow qualifies if it can satisfy the following conditions:

C.1. The volume of traffic routed over $l_{\max}$ is not null.

C.2. The flow is not marked as *modified*.

C.3. Traffic can be removed from at least one topology.

C.4. Traffic can be added to at least one topology.

Conditions *C.3.* and *C.4.* are satisfied if the flow is routed over $l_{\max}$ in at least one topology but not all topologies, i.e. there exists at least one alternative topology in which the traffic flow is not routed over $l_{\max}$. Those flows that satisfy the conditions are then considered iteratively until

the first one that can lead to an acceptable configuration is determined. A new configuration is said to be acceptable if with the new splitting ratios:

C.a. The utilisation of $l_{\max}$ is decreased.

C.b. No link in the network obtains a utilisation higher than the original value of the utilisation of $l_{\max}$.

These ensure that the maximum utilisation is decreased at each iteration and as such, converges to a minimum value that depends on the initial state. If the conditions are satisfied, the new splitting ratios are accepted and reported in the corresponding entries in the DIT. The flow is marked as *modified*, the local view of the DE node about network conditions is updated and the new link $l_{\max}$ is determined. The process terminates when no further local adjustments can be performed. The pseudo-code of the *reconfiguration algorithm* is presented in Algorithm (4.4.1).

**Traffic Splitting Ratio Adjustments** Without loss of generality, let $\mathcal{T}_{l_{\max}}$ represent the set of topologies in which a flow $F(sd)$ is routed over $l_{\max}$, and $\overline{\mathcal{T}}_{l_{\max}}$ the set of topologies in which it is not. The splitting ratios of $F(sd)$ are modified so that the ratios for the topologies in $\mathcal{T}_{l_{\max}}$ are decreased by a factor $\delta^-$ while the ratios for the topologies in $\overline{\mathcal{T}}_{l_{\max}}$ are increased by a factor $\delta^+$, i.e.

$$\forall T \in \mathcal{T}_{l_{\max}}, \quad x_{\mathrm{T}}^{\mathrm{upd}}(sd) = x_{\mathrm{T}}^{\mathrm{cur}}(sd) - \delta^- \tag{4.4}$$

$$\forall T \in \overline{\mathcal{T}}_{l_{\max}}, \quad x_{\mathrm{T}}^{\mathrm{upd}}(sd) = x_{\mathrm{T}}^{\mathrm{cur}}(sd) + \delta^+ \tag{4.5}$$

Parameters $\delta^-$ and $\delta^+$ are functions of the volume of traffic shifted away from $l_{\max}$ and the number of topologies in each set. The updated ratios need to satisfy the two following constraints:

$$\sum_T x_{\mathrm{T}}^{\mathrm{upd}}(sd) = 1 \tag{4.6}$$

$$\forall T, \quad 0 \leq x_{\mathrm{T}}^{\mathrm{upd}}(sd) \leq 1 \tag{4.7}$$

Equation (4.6) ensures that the sum of the splitting ratios is equal to 1 and equation (4.7) guarantees that the ratios are between 0 and 1.

One of the challenges addressed by the *reconfiguration algorithm* is to determine the volume of traffic that can be diverted from $l_{\max}$ in each iteration, while at the same time preserving the network stability. If too much traffic is shifted, other links may become overloaded. This may cause oscillations as in the next iteration traffic will need to be removed from these links.

**Algorithm 4.4.1** Pseudo-code of the reconfiguration algorithm.

**Input**

Local view about the network state

Set of local traffic flows $\phi_{\text{DE}}$ with nbFlows number of flows in $\phi_{\text{DE}}$

**Pseudo-code**

```
nbModifiedFlows ← number of flows in φ_DE marked as modified
adjustmentsArePossible = true
while nbModifiedFlows ≤ nbFlows and adjustmentsArePossible = true do
  l_max ← l such that u(l) = u_max
  φ'_DE ← flows in φ_DE that can be diverted from l_max
  if φ'_DE non null then
    f ← first flow in φ'_DE
    allFlowsAnalysed = false
    canContinue = true
    while allFlowsAnalysed = false and canContinue = true do
      Try to adjust the splitting ratios of f
      if adjustment of f is possible then
        Update entries in the DIT
        nbModifiedFlows++
        canContinue = false
      else
        if next flow in φ'_DE is non null then
          f ← next flow in φ'_DE
        else
          allFlowsAnalysed = true
          adjustmentsArePossible = false
        end if
      end if
    end while
    Update local view about the network state
  else
    adjustmentsArePossible = false
  end if
end while
```

In contrast, if too little traffic is shifted, this may take a long time for the algorithm to converge. The volume of traffic $v_{\text{sd}}$ that can be diverted at each iteration is therefore defined as the total traffic load from the flow on link $l_{\text{max}}$, noted $\rho_{\text{sd}}(l_{\text{max}})$, divided by a factor $2^{\text{K}-\text{k}}$, where $k$ is an integer that varies between 0 and an upper bound $K$ (a parameter of the algorithm), i.e.

$$v_{\text{sd}} = \frac{\rho_{\text{sd}}(l_{\text{max}})}{2^{\text{K}-\text{k}}} \tag{4.8}$$

The value of $k$ is initially set to 0 and is iteratively incremented by 1 until violation of at least one of the traffic shifting conditions (conditions *C.a.* and *C.b.* described in the previous paragraph). As such, the volume of traffic to consider is exponentially increased, which allows a better control of the adjustment steps. The volume shifted from $l_{\text{max}}$ is equally diverted from the topologies in $\mathcal{T}_{l_{\text{max}}}$ and equally distributed across the topologies in $\overline{\mathcal{T}}_{l_{\text{max}}}$[1]. By noting $|\mathcal{T}_{l_{\text{max}}}|$ the size of set $\mathcal{T}_{l_{\text{max}}}$ and $\left|\overline{\mathcal{T}}_{l_{\text{max}}}\right|$ the size of set $\overline{\mathcal{T}}_{l_{\text{max}}}$, the parameters $\delta^-$ and $\delta^+$ can be deduced as follows:

$$\delta^- = \frac{v_{\text{sd}}}{|\mathcal{T}_{l_{\text{max}}}|} \tag{4.9}$$

$$\delta^+ = \frac{v_{\text{sd}}}{\left|\overline{\mathcal{T}}_{l_{\text{max}}}\right|} \tag{4.10}$$

### 4.4.3 Time Complexity Analysis

In order to be implemented, the *reconfiguration algorithm* should be lightweight in terms of computational overhead (i.e. time complexity) imposed at each network edge node. The time complexity of the algorithm is dominated by the number of locally originated flows to consider at each iteration, which depends on the size of the network. In the case of a Point of Presence level topology with $N$ nodes, for instance, where there are traffic demands between any pair of nodes in the network, the number of local traffic flows is $N-1$. The maximum number of local flows for which adjustments can be applied is equal to $N-1$. In the worst case, the maximum number of flows to analyse at each iteration is also equal to $N-1$. As such, the time complexity of the reconfiguration algorithm is $O(N^2)$. Furthermore, it can be noted that the methods used to process the local flows are lightweight in terms of computation, and as such, this requires negligible CPU computation. The actual cost of the adaptation is related to the communication overhead, which is affected by the structure used to connect nodes in the management substrate. The influence of the management substrate on the performance of the approach is discussed in the next section.

[1] Although this may prevent the algorithm from obtaining an optimal solution, it reduces of the complexity of the problem.

## 4.5 Coordination Models for Adaptive Resource Management

### 4.5.1 Full-Mesh Management Substrate Structure

As explained in section 4.3.4, the decision-making process is distributed across the network edge nodes that coordinate through the management substrate (MS) in order to select the Deciding Entity (DE) at each iteration of the adaptation process. This section presents the coordination mechanism in case of the full-mesh structure as described in chapter 3. In this model, network edge nodes are logically connected in a full-mesh fashion, so that each node can logically communicate with every other node in the substrate. Each MS node maintains information about the substrate in local structures managed by the *Management Substrate Information* component as shown in figure (4.3). This contains information about the identities of other nodes in the substrate (e.g. addresses), as well as parameters used for the DE selection.

**Deciding Entity Selection Metric** To prevent inconsistencies between concurrent traffic splitting adjustments, a unique DE is selected at each iteration of the adaptation process. The selection mechanism relies on the value of a *score* metric defined for each MS node and reported to every other node in the substrate.

As explained in section 4.4.2, the objective of the reconfiguration algorithm is to successively adjust the splitting ratios of some of the local traffic flows so that traffic can be diverted from the link with the maximum utilisation in the network. In order to identify the local flows that can potentially be diverted from a link $l$, both static and dynamic characteristics of the flows are considered by the algorithm. The static characteristics are defined according to the configuration of the virtual planes. More specifically, a flow can potentially be diverted from a link $l$ if conditions C.3. and C.4., defined in section 4.4.2, are satisfied, i.e. there exists at least one alternative topology in which the traffic flow is not routed over link $l$. Dynamic characteristics can also be taken into account. A flow is considered as a potential candidate for adjustments if it actually contributes to the load over $l$, i.e. the volume of traffic routed over the link is not null. The static and dynamic characteristics can be translated into two binary variables, noted $m^{\mathrm{S}}_{\mathrm{sd}}(l)$ and $m^{\mathrm{D}}_{\mathrm{sd}}(l)$ respectively, defined for each traffic flow $F(sd)$ and for each link $l$ as follows:

$$m^{\mathrm{S}}_{\mathrm{sd}}(l) = \begin{cases} 0 & \text{if link l is used by flow F(sd) in all topologies or no topology} \\ 1 & \text{otherwise} \end{cases}$$

$$m^{\mathrm{D}}_{\mathrm{sd}}(l) = \begin{cases} 0 & \text{if the volume of flow F(sd) routed over link l is null} \\ 1 & \text{otherwise} \end{cases}$$

Based on the variables $m^{\mathrm{S}}_{\mathrm{sd}}(l)$ and $m^{\mathrm{D}}_{\mathrm{sd}}(l)$, a metric, noted $m_{\mathrm{sd}}(l)$, that indicates whether a traffic flow $F(sd)$ can be considered as potential candidate for traffic removal from link $l$ is

then computed as follows:

$$m_{\text{sd}}(l) = m_{\text{sd}}^{\text{S}}(l) \cdot m_{\text{sd}}^{\text{D}}(l) \tag{4.11}$$

The metric $m_{\text{sd}}(l)$ is equal to 1 if traffic flow $F(sd)$ can be considered as a potential candidate for traffic removal from link $l$, and 0 otherwise. The metric consists of two factors: the static factor $m_{\text{sd}}^{\text{S}}(l)$ and the dynamic factor $m_{\text{sd}}^{\text{D}}(l)$. The value of each factor can easily be computed through information maintained in the LIT and DIT tables (section 4.3.5).

In order to select a unique DE, the definition of the metric is extended at the node level, so that for each node $i$ in the substrate, the *score* of the node, noted $m_{\text{i}}$, is defined as the sum of the metrics $m_{\text{sd}}(l)$ over all links and all local traffic flows averaged by the total number of links and total number of traffic flows (i.e. source-destination pairs) in the network, i.e.

$$m_{\text{i}} = \frac{\sum_{l \in \mathcal{L}} \sum_{F(sd) \in \phi_{\text{i}}} m_{\text{sd}}(l)}{L \cdot \Phi} \tag{4.12}$$

where $L$ and $\Phi$ represents the total number of links and total number of traffic flows in the network respectively. These are static parameters that can be provided in an offline fashion to each of the substrate nodes. The *score* of a node provides an indication of the average number of local traffic flows that can be considered as potential candidate for traffic diversion. The higher the value of the score is, the higher is, on average, the number of potential local traffic flows. The *score* is computed by each MS node locally and communicated to the other nodes through the substrate, so that each node has a global view about the *score* of all its neighbours. The different *scores* are maintained in the *Management Substrate Information* component implemented in each substrate node. Two cases were investigated in this work. In the first case, the proposed metric depends on the static factor only (i.e. the value of the dynamic factor is set to 1). As such, a flow is considered as a potential candidate for diversion based on its static characteristics only. In the second case, both the static and dynamic factors are taken into account to compute the value of the metric $m_{\text{sd}}(l)$.

**Deciding Entity Selection Mechanism** The successive Deciding Entity (DE) nodes are selected at each iteration of the adaptation process based on the decreasing order of the value of their *score*[2], so that the node with the highest score is initially selected. Upon being selected as the current DE, the relevant substrate node executes locally the reconfiguration algorithm that adjusts the splitting ratios of as many local flows as possible so that the maximum utilisation in the network can be decreased. Once the reconfiguration algorithm terminates, the current DE

[2]In the case where the same value is obtained by more than one node, the node identifiers are used to apply the ordering.

**Main actions executed by the DE node**

i. If the DE node has the highest score, then reinitialise the negative reconfiguration number counter.
ii. Execute the reconfiguration algorithm.
iii. If the outome of the algorithm is negative, then increment the negative reconfiguration number counter.
iv. Increment the iteration number counter.
v. If the values of the two counters are smaller than the threshold values, then send a delegation message to next DE node.

Figure 4.4: Main actions executed by the DE node in case of the full-mesh structure.

is responsible for deciding whether to delegate the reconfiguration task, i.e. to select a new DE and continue the adaptation process.

The delegation decision depends on the two following counters: the iteration number counter and the negative reconfiguration number counter. The iteration number counter is initialised at the beginning of the adaptation process by the first DE and is incremented by each successive DE. It is used to monitor the number of iterations of the adaptation process. The negative reconfiguration number counter is initialised by the DE with the highest score. It is incremented by the DE node only in the case where no local adjustment is performed. In order to determine whether to continue the adaptation process, the current DE analyses the value of each counter. The adaptation process terminates if (i), the value of the iteration number counter is equal to the maximum number of permitted iterations (which is a parameter of the system), or (ii), the value of the negative reconfiguration number counter is equal to the number of substrate nodes (this indicates that none of the MS nodes can further improve the performance). The process is continued otherwise. In that case, the current DE node sends a *delegation* message to the next node in the list with the updated values of the counters. In case the current DE is the node with the lowest score, the message is sent to the node with the highest score that reinitialises the value of the negative reconfiguration number counter. The main actions executed by each DE are summarised in figure (4.4).

In the case where only the static factor is used to determine the score, the order of nodes can be precomputed in an offline fashion and provided to each node in the substrate. While this can limit the volume of messages that need to be exchanged between nodes in the substrate, this does not take into account the evolution of the characteristics of the flows. This may lead to an increase in the number of negative reconfiguration (i.e. cases where no local adjustment can be performed) and as such, may affect the convergence of the approach. In contrast, the

score of each node is reevaluated at the end of each adaptation cycle in the case of the dynamic score. In order to obtain the new ordered list of nodes, the last DE of the adaptation cycle is responsible for sending a score computation request to all nodes in the substrate. Upon receiving the request, the nodes recompute the value of their score and forward the updated value to all their neighbours so that the ordered list can be recomputed independently by each node locally. The influence of the DE selection method on the overall performance of the proposed scheme is evaluated in the section 4.6.

### 4.5.2 Hybrid Management Substrate Structure

As explained in chapter 3, in the case of the full-mesh model, each node in the substrate needs to maintain information about every other substrate node, which may raise some limitations as the number of nodes increase. In this section, the use of the hybrid model presented in chapter 3 to connect the network edge nodes is investigated. In the hybrid model, network edge nodes are partitioned into a set of clusters and nodes in each cluster are connected according to a ring topology. The different sub-rings are then inter-connected in a fully-meshed fashion through Intermediate Entity (IE) nodes so that there exists exactly one IE node in each sub-ring. Communication between the different sub-rings is performed through the IE nodes so that each IE acts as an interface to the other sub-rings. Each IE maintains information about all nodes in its cluster and about other IE nodes locally in the *Management Substrate Information* component. In contrast the other nodes maintain locally information about their direct neighbours in the ring only.

**Sub-ring Score Metric** While the selection of the deciding entity is done at the node level in the case of the full-mesh approach, the selection mechanism used in the hybrid model is applied at the sub-ring level. More specifically, in a similar fashion to the full-mesh model, the *score* of each node in each sub-ring is computed according to equation (4.12). These are then used to compute the *score* of each sub-ring $R$ as the sum of the *score* of each node in the sub-ring averaged by the total number of nodes in the sub-ring, i.e.

$$M_{\mathrm{R}} = \frac{\sum_{i \in R} m_{\mathrm{i}}}{|R|}. \tag{4.13}$$

where $|R|$ represents the size of the sub-ring $R$. The metric $M_{\mathrm{R}}$ is computed by the $IE$ node in the sub-ring $R$ based on the value of the *score* reported by each node in the ring. The value is then provided to every other $IE$ node, so that the order list of sub-rings can be computed locally by each IE node. In a similar fashion to the previous model, two cases were considered in this work: a) the case where static parameters only are used to compute the score and b) the case where dynamic parameters are also considered.

**Main steps of the sub-ring selection mechanism**

i. Select the sub-ring with the highest score.
ii. Execute the reconfiguration process in the selected sub-ring.
iii. Check conditions to continue the adaptation process.
iv. If conditions to continue are satisfied, select the next sub-ring in the list and go back to step (ii). Otherwise, terminate.

Figure 4.5: Main steps of the sub-ring selection mechanism.

**Sub-ring Selection Mechanism** The sequence of reconfigurations performed during the adaptation process is executed at two levels. Based on the value of the *score* of each sub-ring, each IE node computes locally an ordered list of sub-rings, so that these are ranked according to the decreasing value of their *score*[3]. The sub-rings are then iteratively considered according to their rank in the list. Once a sub-ring is selected, the relevant IE node triggers a sequence of reconfigurations within the ring according to a hop-by-hop mechanism where reconfiguration decisions are successively made by each node in the cluster. When the process terminates, the IE verifies that the conditions to continue the adaptation process are satisfied, and in that case, delegates the reconfiguration task to the next sub-ring in the list. The main steps of the sub-ring selection mechanism are summarised in figure (4.5).

In order to control the sequence of reconfigurations, two counters are considered by the IE nodes: a) the iteration number counter and b) the negative reconfiguration number counter. The iteration number counter is used to control the number of iterations of the adaptation process and is incremented by each node involved in the process. The negative reconfiguration number counter is defined at the sub-ring level and is used by the IEs to determine whether further adjustments can be performed with another sub-ring. It is (re-)initialised by the IE in the sub-ring with the highest *score* and it is incremented by an IE in the case where no adjustment could be performed in its sub-ring. The values of the two counters are provided by the current IE to the next IE in the delegation request message sent in order to select the next sub-ring. The adaptation process terminates if the value of the iteration number counter is equal to the maximum number of permitted iterations or if the value of the negative reconfiguration number counter is equal to the number of sub-rings.

**Sub-ring Reconfiguration Process** The reconfiguration process which is triggered in each sub-ring works as follows. Initially, the IE executes locally the reconfiguration algorithm. Once the

[3]In the case where more than two rings have the same score, the ranking is decided based on the identifier of the IE nodes.

algorithm terminates, the IE increments the value of the iteration number counter and analyses the updated value to decide whether to continue the process. In that case, it reports the value of the counter as well as the outcome of the algorithm (i.e. positive in case of successful adjustments and negative otherwise) in a delegation request message which is sent to its neighbour node. The process stops in the case where the value of the iteration number counter is equal to the maximum number of permitted iterations. Upon receiving the request message, the next hop node executes locally the reconfiguration algorithm, increments the iteration number counter and analyses the updated value. If it can continue, it appends the outcome of the algorithm to the delegation request message which is then forwarded to the next hop node. Once the message reaches the IE, it is analysed, and, if no successful reconfiguration is reported (i.e. no feasible adjustment was performed in the ring), the value of the negative reconfiguration number counter is incremented.

## 4.6 Evaluation

This section presents the results of the evaluation of the different mechanisms of the proposed adaptive resource management approach. It first investigates the performance in terms of resource utilisation that can be achieved with the load balancing scheme, and it then evaluates the influence of the coordination model.

### 4.6.1 Experimentation Setup

**Java Software** In order to implement the proposed approach, a Java-based software was developed. The software consists of a set of functionalities that enables the computation of relevant network statistics (i.e. link load and utilisation) given a network topology, specific traffic engineering settings (e.g. traffic splitting ratios) and input traffic matrices. More specifically, the program relies on three main functionalities defined as follows:

- the monitoring functionality to compute network statistics.
- the adaptation functionality to control and execute the adaptation process.
- the reconfiguration functionality to adjust the splitting ratios of traffic flows according to the reconfiguration algorithm.

The software is designed so that several inputs can be specified by the user: (i) the physical network topology, for which the set of edge nodes is defined, (ii) the structure of the management substrate used to connect the edge nodes, which is computed based on the methods presented in chapter 3, (iii) the set of traffic matrices related to the incoming demand at the

network edge nodes at specific time intervals over a predefined time period, (iv) the number of virtual topologies to consider, (v) the maximum number of iterations of the adaptation process, and (vi) the total time period to consider.

Based on the inputs, the program computes the set of virtual topologies according to the algorithm described in section 4.3.3. These are then used to extract the list of paths between each source-destination pair of nodes in the network. The input traffic matrices are considered iteratively according to the specified time period. For each traffic matrix, the program executes several steps. It first invokes the monitoring functionality to compute the network statistics for the given time interval and the current traffic splitting ratios. Based on these, it retrieves specific information such as the link with the maximum utilisation in the network. This is passed to the adaptation functionality component that executes the adaptation process according to the management substrate model considered. At each iteration, it determines the relevant Deciding Entity edge node according to the selection mechanism associated with the substrate. Upon selecting the Deciding Entity, it calls the reconfiguration functionality that executes the reconfiguration algorithm based on the attributes of the corresponding edge node. The program emulates the actual communication between the different nodes in the substrate through read and write methods applied to the structures instantiated for each network edge node.

A traffic engineering tool related to the developed software was previously proposed in the literature [102][103]. The tool, called TOTEM (TOolbox for Traffic Engineering Methods), provides a set of methods to evaluate the performance of different routing schemes in terms of network resource utilisation. Despite the availability of several routing mechanisms (e.g. Shortest Path First algorithm, Interior Gateway Protocol-Weight Optimization algorithm etc.), the tool lacks of flexibility to integrate the new functionalities required for the proposed adaptive scheme.

To guarantee the integrity of the different functionalities of the software, the required testing procedures have been applied the different methods and classes. In addition, the Shortest Path First algorithm was implemented and the results in terms of resource utilisation were compared against those obtained with the TOTEM toolbox. The experiments were performed on a laptop with a 2.80 GHz Intel Core i7-2640M processor and 8 GB memory.

**Network Topologies and Traffic Datasets** The performance of the proposed approach is evaluated on the two Point of Presence (PoP) level topologies considered in chapter 3, namely the Abilene network [99] and the GEANT network [100], for which real traffic measurement datasets, as well as link capacities and weights, are available. The Abilene network consists of 11 PoP (or nodes) and 28 unidirectional links. 26 links have a capacity of 9.2Gbps and 2

| | GEANT | Abilene |
|---|---|---|
| **2 topologies** | 89.4% | 73.3% |
| **3 topologies** | 13.2% | 20.0% |
| **4 topologies** | 0% | 0% |

Table 4.3: Percentage of critical links according to the number of topologies.

links have a capacity of 2.8Gbps. The GEANT network considered in this work consists of 21 PoP and 68 unidirectional links. 32 links have a capacity equal to 10Gbps, 28 links equal to 2.4Mbps, 4 links equal to 622Mbps and 4 links equal to 155 Mbps.

In order to represent a wide range of traffic conditions, traffic matrices over a period of 7 days are considered in the evaluation. Traffic traces for the Abilene network are available at 5 minute intervals [99], while they are provided at the timescale of 15 minute intervals in the case of GEANT [93]. In order to maintain the consistency between the two networks, adaptation is performed at a frequency of 15 minutes in both cases. This time period was appropriate for the experiments so that traffic fluctuations could be followed without introducing frequent configuration changes[4]. As a result, a set of 672 traffic matrices was considered. Each traffic matrix is noted $TM_{\mathrm{i}\in[1;M]}$ where $M$ is the total number of traffic matrices to consider.

### 4.6.2 Virtual Topologies Settings

The virtual topologies are computed according to the algorithm presented in section 4.3.3. Table (4.3) indicates for each network the percentage of links that cannot satisfy the path diversity requirements (i.e. critical links) according to the number of virtual topologies computed. The optimum number of virtual topologies, noted OPT-VT, is defined as the number of topologies required so that all links in the network can satisfy the path diversity requirements, i.e. there is no critical link. In both Abilene and GEANT, the optimum is equal to four. This is consistent with [123] that argues that a small number of topologies (typically a maximum of five topologies) is required to satisfy all links.

### 4.6.3 Load Balancing Scheme Performance

The objective of the experiments is to evaluate the performance in terms of resource utilisation that can be achieved with the proposed adaptive load balancing approach. In order to evaluate the actual gain, three different schemes were considered:

- **Original scheme**: the original link weight settings are used in the original topology and no adaptation is performed

[4]Experiments were also performed by considering 5 minute and 30 minute intervals. The results showed that the choice of the time interval does not significantly affect the performance.

- **Load Balancing (LB) scheme**: the proposed approach; virtual topologies are used to provide path diversity and the traffic splitting ratios are periodically adjusted. The traffic between any source-destination pair is initially distributed evenly across the different available paths.

- **The optimum**: the routing problem is defined as a Multicommodity Flow problem [97] and the *glpsol* GLPK (GNU Linear Programming Kit) linear programming solver [101] is used to compute the optimal maximum utilisation for each traffic matrix.

The maximum utilisation in the network (max-u) obtained with each scheme at each measurement period (i.e. for each traffic matrix) was determined and the performance achieved with the Original scheme and the LB scheme were compared against the optimum. Online traffic engineering approaches proposed in the literature achieve near-optimal performance (e.g. [73][71][74]). Instead of choosing an existing algorithm to compare against, it is believed that directly comparing to the optimum can provide the most relevant performance indicator. The max-u obtained with the optimum scheme is noted $u^*_{\text{max}}$. In order to compare the performance of the Original scheme and the LB scheme, three evaluation factors are defined as follows:

- **Average deviation (AD)**: this indicates the average deviation of the max-u from the optimum over the set of measurements (i.e. for all traffic matrices) as follows:

$$AD = \frac{\sum_{i=1}^{M} \left( \frac{u_{\text{max}}}{u^*_{\text{max}}} \right)_{\text{TM}_\text{i}}}{M} \tag{4.14}$$

- **Percentage Near-Optimal Max-u (PNOM)**: this indicates the percentage of traffic matrices for which near-optimal performance can be achieved. Near-optimal performance is defined so that the obtained max-u is within 10% of the optimum. Let $\pi_\text{i}$ be the binary variable defined as follows:

$$\forall i \in [1; M] \quad \pi_\text{i} = \begin{cases} 1 & \left( \frac{|u_{\text{max}} - u^*_{\text{max}}|}{u^*_{\text{max}}} \right)_{\text{TM}_\text{i}} \leq 0.1 \\ 0 & \text{otherwise} \end{cases} \tag{4.15}$$

  The factor PNOM is then defined as follows:

$$PNOM = \frac{\sum_{i=1}^{M} \pi_\text{i}}{M} \cdot 100 \tag{4.16}$$

- **Highest Maximum Utilisation (HMU)**: this indicates the highest max-u obtained in the network across all the traffic traces during the considered time period.

$$HMU = \max_{\text{i} \in [1;M]} (u_{\text{max}})_{\text{TM}_\text{i}} \tag{4.17}$$

| | GEANT | | | Abilene | | |
|---|---|---|---|---|---|---|
| | **AD (%)** | **PNOM (%)** | **HMU (%)** | **AD (%)** | **PNOM (%)** | **HMU (%)** |
| **Optimum** | - | - | 53.79 | - | - | 12.19 |
| **Original** | 89.88 | 0 | 83.09 | 54.34 | 0 | 21.16 |
| **LB_FM_S** | 7.15 | 96.14 | 54.83 | 7.53 | 94.82 | 12.87 |
| **LB_FM_D** | 9.18 | 95.90 | 56.12 | 6.16 | 96.91 | 13.67 |
| **LB_H_S** | 8.74 | 95.14 | 55.44 | 8.72 | 94.96 | 13.31 |
| **LB_H_D** | 8.29 | 95.43 | 54.51 | 8.93 | 95.24 | 13.10 |

Table 4.4: Performance obtained with the different schemes.

In order to investigate the influence of the management substrate on the performance of the LB scheme, the four following scenarios were considered: the full-mesh topology with static score (FM_S), the full-mesh topology with dynamic score (FM_D), the hybrid topology with static score (H_S) and the hybrid topology with dynamic score (H_D). In all experiments, the number of virtual topologies is equal to $4$ and the maximum number of permitted iterations is fixed to $50$. The traffic load is initially equally balanced across the different virtual planes. The results are summarised in Table (4.4).

As can be observed, the performance of the LB schemes is not affected by the management substrate structure. Near-optimal performance is obtained in all scenarios in both the GEANT and Abilene networks, with an average deviation from the optimal less than $10\%$. In addition, the PNOM factor shows that the proposed scheme performs uniformly well given that near-optimal results are obtained for more than $95\%$ of the traffic matrices. In contrast, the Original scheme achieves poor performance in terms of resource utilisation compared to the Optimal scheme. The average deviation from the optimum is equal to $54\%$ in the case of Abilene and reaches almost $90\%$ in the case of GEANT. The HMU factor shows that the max-u in the network can be significantly reduced with the LB scheme. The evolution of the max-u, obtained with the LB scheme (FM_S case) and the Original scheme, at 15 minute intervals in the case of the Abilene and GEANT networks is plotted in figure (4.6) and figure (4.7), respectively. As it can be observed, the LB scheme can achieve a significant gain in terms of resource utilisation compared to the Original scheme.

The performance of the proposed load balancing scheme can also be affected by the underlying routing configuration, i.e. by the virtual planes. As shown in section 4.6.2, four virtual topologies are required in both Abilene and GEANT to satisfy the path diversity requirements (section 4.3.3). In this paragraph, the influence of the number of virtual topologies (NbT) on the performance of the LB scheme is investigated. The performance obtained in the case of

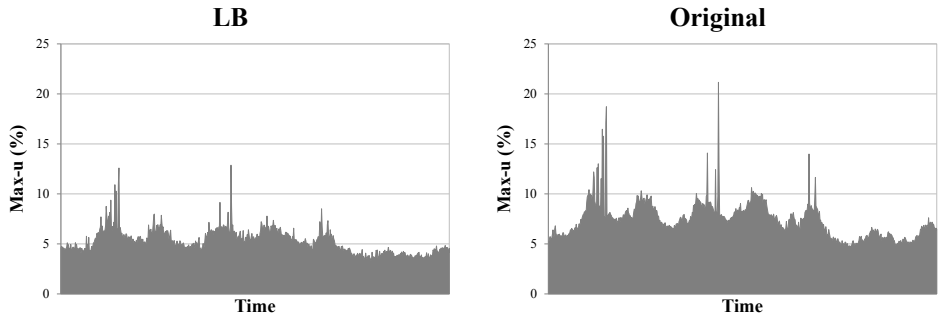



Figure 4.6: Evolution of the maximum utilisation at 15 minute intervals using (a) the Load Balancing (LB) scheme, and (b) the Original scheme, for the Abilene network.

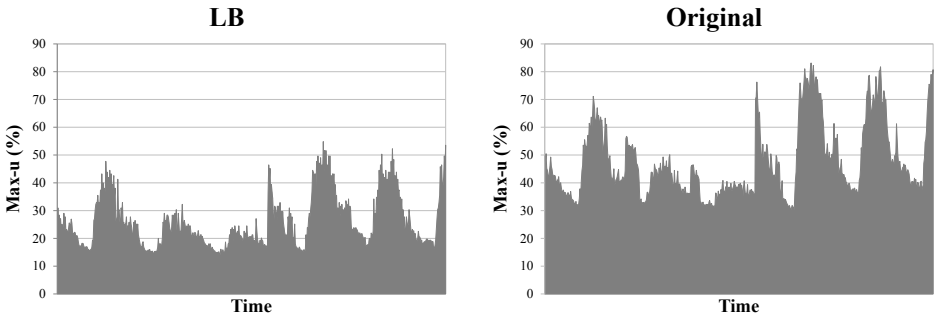



Figure 4.7: Evolution of the maximum utilisation at 15 minute intervals using (a) the Load Balancing (LB) scheme, and (b) the Original scheme, for the GEANT network.

| | GEANT | | | Abilene | | |
|---|---|---|---|---|---|---|
| | **AD (%)** | **PNOM (%)** | **HMU (%)** | **AD (%)** | **PNOM (%)** | **HMU (%)** |
| **Optimum** | - | - | 53.79 | - | - | 12.19 |
| **Original** | 89.88 | 0 | 83.09 | 54.34 | 0 | 21.16 |
| **NbT = 2** | 90.89 | 0 | 86.03 | 30.82 | 9.35 | 18.79 |
| **NbT = 3** | 16.45 | 47.77 | 57.04 | 10.22 | 85.50 | 15.04 |
| **NbT = 4** | 7.15 | 96.14 | 54.83 | 7.53 | 94.82 | 12.87 |

Table 4.5: Influence of the number of virtual topologies.

the FM_S scenario with different number of routing planes is reported in Table (4.5). Similar results were obtained with the three other scenarios. For comparison purposes, the performance of the Original scheme and the Optimum are also indicated in the table.

As it can be noted, the performance of the LB scheme improves as the number of virtual topologies increases. The best performance is obtained in both networks when the optimal number of topologies is used to provide path diversity (i.e. all links satisfy the path diversity requirements). While a gain in terms of resource utilisation can be achieved with two virtual planes in the case of the Abilene network (AD of around $30\%$), the performance obtained with two virtual planes are similar to the performance of the Original scheme in the case of GEANT. In this case, given that some links do not satisfy the path diversity requirements (section 4.3.3), it may not be possible to remove traffic from some of the links in the network and, as such, the algorithm has less flexibility to balance the traffic load. The results show that a small number of routing topologies is enough to achieve near-optimal performance, which is encouraging regarding the scalability and applicability of the proposed approach.

Finally, it should be noted that the results obtained in the case of the Abilene and GEANT topologies, not only depend on the characteristics of the proposed load balancing algorithm but also on the method used to compute the virtual routing planes. In particular, the satisfaction of the path diversity requirements (section 4.3.3) is essential to obtain the observed performance since this directly affect the flexibility of the algorithm to divert the traffic load in the network.

### 4.6.4 Management Overhead and Complexity Analysis

In the previous section, it was shown that the performance in terms of resource utilisation of the load balancing scheme is not affected by the management substrate. However, as explained in section 4.4.3, the substrate structure may influence the management cost and complexity (i.e. number of messages and number of iterations) of the proposed approach.

The influence of the maximum number of permitted iterations on the performance of the load balancing scheme is investigated for the four scenarios considered in the case of the LB

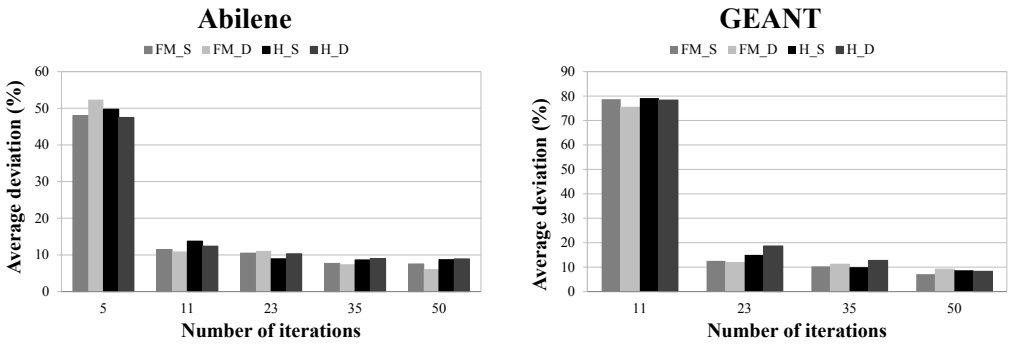



Figure 4.8: Influence of the number of iterations.

scheme. The evolution of the average deviation of the maximum utilisation according to the maximum number of permitted iterations is plotted for each scenario and each network in figure (4.8).

As it can be observed, the four scenarios follow similar trends. In all cases, the average deviation decreases as the maximum number of iterations increases. Due to the characteristics of the adaptation process, the convergence of the algorithm is affected by the initial configuration of the splitting ratios. In the case where the demand does not change significantly between two consecutive monitoring intervals, no major modification of the settings is required to re-balance the traffic load. As such, new splitting ratios can be computed in a small number of iterations only. In contrast, a larger number of iterations may be required in case of dramatic changes in the demand. In the case of the Abilene network, a minimum of $11$ iterations is required to obtain an average deviation of less than $15\%$, while $23$ iterations are necessary in the case of GEANT. This is influenced by the number of traffic flows to reconfigure in order to re-balance the traffic load, and as a result, by the number of nodes in the substrate. The results show that the method used to select the Deciding Entity node does not affect the convergence of the load balancing algorithm.

The selection mechanism can, however, affect the management overhead (i.e. number of messages) incurred by the communication between the substrate nodes. In the case of the dynamic selection, the score of each node needs to be recomputed and communicated to every node in the substrate at each iteration of the adaptation process. The communication overhead is therefore more significant than in the static selection case. Given that the objective of the adaptation process is to iteratively improve the performance in terms of resource utilisation (i.e. to reduce the maximum utilisation), the performance of a selection method can be measured based on the number of iterations at which a negative outcome is obtained (i.e. no feasible local reconfiguration). In that case, this leads to an unnecessary use of communication and

| | GEANT (%) | Abilene (%) |
|---|---|---|
| **LB_FM_S** | 4.44 | 8.81 |
| **LB_FM_D** | 5.98 | 10.08 |
| **LB_H_S** | 5.05 | 10.12 |
| **LB_H_D** | 4.43 | 10.18 |

Table 4.6: Percentage of the number of iterations with a negative outcome.

computation resources. The percentage of the number of iterations with a negative outcome obtained with each selection method is presented in Table (4.6). The total number of iterations is fixed at 50. The lower the percentage is, the better the selection method is considered to be.

No significant difference can be observed between the different methods. The percentage is on average equal to 10% in the case of Abilene and to 5% in the case of GEANT, i.e. only a small number of iterations is associated with a negative outcome. The results show that the use of the dynamic score to rank the nodes does not lead to any performance improvement. By design, this method is more costly in terms of management overhead than the static one. As such, it can be concluded that the static model leads to better performance. Furthermore, due to the absence of major differences in the results obtained with the full-mesh and the hybrid models, it can be noted that the structure of the management substrate does not affect the performance of the proposed resource management scheme. The choice of the structure to use is therefore essentially driven by the characteristics of the network topology as described in chapter 3.

Finally, the order of magnitude of the computing time of the reconfiguration algorithm executed by a network node edge to determine new splitting ratios is evaluated. In the proposed implementation, it takes on average less than few milliseconds for a network edge node to execute the reconfiguration algorithm. In order to compare the two networks, the ratio of the average execution time to the total number of local traffic flows was computed in both cases. This is equal to 0.15 in the case of Abilene and to 0.18 in the case of GEANT, which shows that, in practice, the computing time is proportional to the number of local traffic flows.

## 4.7 Conclusion

This chapter presents a new intra-domain resource management approach for IP networks, where the traffic distribution is controlled in an adaptive and decentralised manner according to network conditions. Unlike offline traffic engineering schemes, which rely on static configurations, the proposed approach can efficiently deal with network and traffic dynamics by performing the adaptation of routing configurations in short timescales. New configurations are

not computed by a centralised management entity as it is the case, for instance, in [74]. These are the results of reconfiguration actions performed directly by the network edge nodes. In contrast to previous approaches [108][71], in which load balancing decisions are made by each node in the network, edge nodes only are involved in the reconfiguration process. However, unlike the approach in [73], where the adjustment actions are supported by a control mechanism implemented by the core nodes, edge nodes coordinate among themselves through the management substrate described in chapter 3 to decide upon the most appropriate course of actions to follow in order to re-balance the traffic load. As such, new functionality needs to be deployed in the network ingress nodes only and no modification of the core nodes is required. The benefits of the proposed approach are demonstrated on realistic topologies and traffic traces. The results of the evaluation indicate that near-optimal performance can be achieved in terms of resource utilisation with the proposed load balancing scheme, by using a small number of routing topologies. In addition, they show that the management substrate can efficiently support the reconfiguration of network resources in a responsive and scalable manner.

The proposed adaptive resource management approach is extended in the next chapter for the purpose of energy management.

## Chapter 5

# Energy-Aware Adaptive Network Resource Management

## 5.1 Introduction

Energy awareness has been the subject of technological developments over the past decade, ranging from simple energy-saving techniques for battery-powered computer equipment to more sophisticated ones applying to data centers [126]. The increasing power consumption of modern networks, driven by bandwidth-hungry applications, in conjunction with the rising cost of energy - and therefore operational expenditure (OPEX) - and increasing environmental consciousness, has led researchers to investigate methods by which the carbon footprint of network infrastructures can be reduced. Although some work has been carried out in this area, effective energy management solutions are still missing. Existing approaches in the literature that address energy efficiency in network infrastructures mainly propose offline and centralised solutions which assume the availability of traffic demand, e.g. [77][127]. However, these approaches can have sub-optimal performance under changing or unpredicted traffic conditions and also have inherent scalability limitations. The most prominent method proposed by which energy consumption can be reduced is powering off links/routers [75], since the alternative method of rate adaptation [128] has not demonstrated significant gain. However, switching off entire links/routers can disconnect the network topology, which, in addition to possible packet losses under traffic variations, is not suitable for online reconfigurations given that the process of computing the routing configuration is not trivial. This chapter proposes a new online energy-aware resource management approach which can be seen as a middle ground between the reduced topology and rate adaptation solutions. By exploiting the fact that many links in core networks are bundles of multiple physical cables, the proposed approach adapts the link capacity at run time by switching off individual line cards (LCs) [8]. This is achieved by controlling the traffic splitting ratios at network ingress nodes in a decentralised fashion, which

results in offloading some LCs that can subsequently enter sleep mode. The choice of switching off LCs as the means to save energy was motivated by the real measurements reported in [75]. According to these, the overall power profile of a network is dominated by the energy consumption of the router LCs and the chassis, with an approximate ratio of 3:1 for a fully loaded Cisco GSR 12008.

To achieve the energy-saving objective, the adaptive resource management framework and the intelligent in-network substrate presented in chapter 3 were extended. The substrate is a logical structure formed between the ingress nodes (edge nodes) of a network and encapsulates the necessary logic to realise self-management functionality. Substrate nodes coordinate reconfiguration actions that control the traffic distribution in the network. Given that the optimal mapping of traffic demand to available resources is an NP-hard problem, and also the strict time constraints of an online reconfiguration process, such as the one proposed here, an efficient heuristic algorithm was devised. This is executed by substrate nodes periodically (every 15 minutes) and produces a new configuration in the form of traffic splitting ratios. In a similar fashion to the load balancing application presented in chapter 4, the proposed approach is based on the path diversity provided by multi-topology routing (MTR), with the traffic volume on paths between source-destination pairs being defined by the computed ratios. The approach was evaluated using real topologies and traffic traces, and its performance was compared against that of a load balancing algorithm (i.e. the approach presented in chapter 4) and plain MTR (i.e. no adaptation). The results indicate that substantial energy gain can be achieved without significantly compromising the balance of the network in terms of load.

The rest of the chapter is organised as follows. Section 5.2 discusses related work. Section 5.3 introduces the link model considered in this work and provides an overview of the proposed approach. Section 5.4 details the adaptive traffic distribution algorithm. Section 5.5 presents and analyses the results of the evaluation of the approach using the Abilene and GEANT network. Finally, the main contributions of this work are summarised in section 5.6.

## 5.2 Related Work

Resource management in fixed networks is recently adopting the objective of energy efficiency, attempting to better associate network conditions with energy consumption. The common strategy of resource over-provisioning with constant energy consumption will be gradually replaced by new adaptive and deployable approaches, balancing performance with energy efficiency. Proposed solutions are based on various assumptions for the network environment. They consider networks with bundled links, links with single LCs or going deeper into the level of LC

ports. A bundled link could be a fiber and its set of WDM channels as in [129], or other types of fixed physical links. The topology could be dynamic, in the sense that it is associated with reduced connectivity graphs due to strategies that turn off LCs and corresponding links, e.g. [76][130][131]. Other approaches assume fixed topologies but dynamic traffic conditions, at short or longer timescales, e.g. [132][133]. In [129], the authors define theoretical upper bounds in the energy-saving potential for different types of dynamicity in the environment, including topology and routing strategies. Such assumptions could be realistic but are associated with non-trivial complexity.

As explained in chapter 4, the assumptions for the network environment call for the deployment of efficient monitoring mechanisms. While most of the proposals rely on integrated monitoring facilities (e.g. [77][76]), it is believed that a separate monitoring or information management system could be used to satisfy the requirements in terms of network awareness. In a similar fashion to chapter 4, the information management overlay proposed in [83] can be used to provide network information.

An approach that is aware of the network environment conditions should be able to take energy-saving actions in response to such conditions. The type of actions range from pruning the network topology (e.g. exploiting algebraic connectivity [130]), routing changes (e.g. of shortest path tree [76][134][135]), to traffic engineering approaches (e.g. setting traffic split ratio among multiple paths [132][133]). The traffic engineering approaches usually use MPLS tunnels or virtual network links (e.g. [133][123]). Furthermore, the majority of the proposed solutions take actions in an offline and centralised manner (e.g. [130][127][134][135]). Very few of them have online considerations, e.g. [132] that proposed a centralised energy-aware traffic engineering approach for MPLS networks, or [76][136] that focused on distributed approaches for MPLS-based and OSPF-based networks, respectively. In [76], energy-saving is achieved through a mechanism, deployed at the ingress nodes of the network, which sequentially recomputes the paths between any source-destination pair of nodes and adapts the reserved capacity on each link. In contrast, the mechanism developed in [136] is distributed over all network nodes. These independently decide whether to turn on or off the local interfaces. Other approaches targeting these issues are still at an early stage [133][134]. Still other approaches adopt solutions belonging in the middle ground between the offline and online solutions; for example using topology switching [77][137]. In that case, a set of topologies are pre-configured according to the expected network conditions at certain times of the day and the choice of the selected topology is driven by the current time period. Another characteristic of these approaches is the switching off strategy considered. Most of the previous work has focused on strategies to

| Approach | Centralised/ Decentralised | Offline/ Online | Switching off strategy | Energy-saving action |
|---|---|---|---|---|
| [130] | centralised | offline | line card | network connectivity adaptation |
| [132] | centralised | online | line card | MPLS-based traffic-engineering |
| [127] | centralised | offline | line card | routing changes |
| [137] | centralised | offline | link | topology switching |
| [77] | centralised | offline | link | topology switching |
| [134] | centralised | offline | link | routing changes |
| [135] | centralised | offline | link | routing changes |
| [76] | decentralised | online | line card | routing changes |
| [133] | decentralised | online | link | MPLS-based traffic engineering |
| [136] | decentralised | online | link | routing changes |

Table 5.1: Characteristics overview of energy-saving network management approaches.

switch off the entire link, e.g. [77][137][134][136]. Some of these approaches (e.g. [127][76]) performed the switching off actions at the line card level, which allows a finer level of control of the traffic distribution in the network.

A summary of the characteristics of the main body of work described in this section is presented in Table (5.1). The table indicates, for each approach, whether it relies on an offline or online mechanism and whether the management process is centralised or decentralised. It also indicates the switching off strategy considered, as well as the type of energy-saving actions.

## 5.3 Energy-aware Dynamic Capacity Adaptation

This section describes the proposed energy-aware resource management approach. Given that the approach is based on adapting the capacity of bundled links, an illustration of link aggregation from an energy consumption viewpoint is first provided.

### 5.3.1 Link Aggregation

Aggregating multiple physical cables into a single logical link to connect IP routers has been standardised in [138] and is common practice in today's core networks. The aggregation (or bundling) capability is a result of the modular architectures of modern routers, where multiple line cards (LCs) can be attached to the switching fabric. Under a single layer 3 logical address, bundled line cards can allow increased throughput beyond the capacity of a single connection, for example that of the fastest available link technology. Link aggregation can also be used as a flexible solution for network capacity upgrades whereby bundles are extended with new LCs instead of replacing existing links with higher-capacity ones.

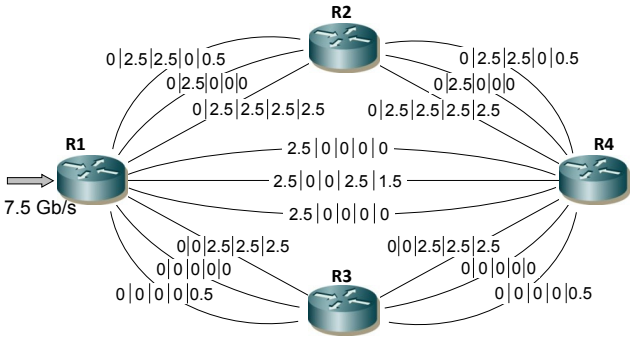



Figure 5.1: Examples of traffic distribution among multiple line cards.

Using a simple topology, figure (5.1) illustrates five different load distributions (separated by '|') for an input traffic volume of 7.5 Gb/s, which is allocated by the source node R1 between three possible paths to the destination node R4. All bundled links consist of three LCs, each with a capacity of 2.5 Gb/s. In this chapter, it is considered that all LCs in a bundle have the same capacity. In addition, multiple ports on individual LCs are viewed as a single interface. Finally, it is assumed that the allocation of traffic on a bundled link is carried out on a fill-first-available LC basis. In contrast to a strategy which balances the traffic load across all available LCs in a bundled link, this ensures that the load is concentrated on the minimum number of LCs required to support the incoming traffic in each bundled link, and, as such, enables non-utilised LCs to be turned off.

Table (5.2) summarises the result of the traffic distribution examples in terms of the number of LCs used, with each unbundled link being associated with two LCs, (one at either end, e.g. egress of R1 and ingress of R4). The least number of LCs (6) used is achieved in case (a), where all traffic is routed through the shortest path (R1-R4) of one hop. The LC number doubles to 12 when using the longest path (R1-R2-R4) of two hops in (b). Splitting the traffic with ratios of 0.75 and 0.25 between the longest paths R1-R2-R4 and R1-R3-R4, respectively in case (c), does not improve the cost compared to (b). However, equal splitting among the three paths in (d) reduces the LCs used by 2 since a third of the traffic is now routed over the shortest path. The last example (e) concerns the case where the load on a bundled link is not a multiple of the LC capacity. As a result, some LCs are not utilised to their full capacity and thus a higher number is required to accommodate the load.

Measuring the energy consumption of a network by the number of LCs used and based on the above, some observations can be made: (i) using shortest paths incurs the least cost, but can lead to congestion; (ii) optimised splitting ratios can reduce energy consumption with

| Router | LCs Available | Line Cards Used | | | | |
|---|---|---|---|---|---|---|
| | | (a) | (b) | (c) | (d) | (e) |
| R1 | 9 | 3 | 3 | 3 | 3 | 5 |
| R2 | 6 | 0 | 6 | 4 | 2 | 4 |
| R3 | 6 | 0 | 0 | 2 | 2 | 4 |
| R4 | 9 | 3 | 3 | 3 | 3 | 5 |

Table 5.2: Line card usage examples.

load balancing mitigating congestion; (iii) non-fully utilised LCs incur a high cost and should be avoided where possible. These observations were the key foundations in the design of the proposed algorithm.

### 5.3.2 Approach Overview

**Notations** In this chapter, it is considered that each network link consists of the aggregation of multiple physical cables and is therefore referred as a bundled link (BL). Let $l \in \mathcal{L}$ denote the set of BLs in the network. Let $\rho(l)$ be the traffic load on each bundled link $l$ and $c(l)$ be its capacity. The later is defined as the sum of the capacity of each LC in the bundled link, where the capacity of each line card in a bundled link $l$ is noted $c_{\text{LC}}(l)$ (as previously stated, it is assumed that all LCs in the bundle have the same capacity). Line cards can be *full* (LCF) if their load is equal to their capacity, *utilised* (LCU) if their load is not zero and less than their capacity, and *non-utilised* (LCN) if they have zero load.

**Proposed Approach** In contrast to the main body of work in energy efficiency of network infrastructures, which propose centralised offline solutions, a flexible approach by which the configuration of a network can be adapted dynamically, in a decentralised fashion, to meet energy efficiency objectives, was designed. To reduce the energy required to sustain the operation of a network, the proposed approach is based on the use of multiple LCs to implement the links between IP routers and the capability of individual line cards to enter sleep mode. To achieve the latter, traffic is distributed in such a way that some of the LCs are not utilised, i.e. no traffic is transmitted through them. This is controlled by dynamically adapting the splitting ratios applied to incoming traffic at the ingress nodes, according to a reconfiguration algorithm. The resulting configuration enables some LCs to sleep, which thus adapts the bundled link capacity. The main advantage of this approach, compared to very recent work in [76] and [139], is that the physical topology does not become disconnected and new source-destination paths do not need to be computed.

The reconfiguration process is managed by the intelligent in-network substrate presented in chapter 3 that was used for load balancing purposes in chapter 4. In a similar fashion to the load balancing approach, the use of a full-mesh and hybrid structure was considered to logically connect participating nodes in the management substrate. As shown in chapter 4, the performance of the resource management scheme is not affected by the management substrate structure, and as such, the choice of the model to use is essentially driven by the characteristics of the physical network topology. The adaptation is performed periodically in short timescales (i.e. every 15 minutes) at which point substrate nodes coordinate their actions for the computation of new traffic splitting ratios in a decentralised manner. Adaptation follows a similar mechanism as the one developed for load balancing in chapter 4. It is an iterative process with only one substrate node permitted to assume the role of the Deciding Entity (DE) at each iteration and subsequently execute the reconfiguration algorithm. The methods used to select the DE are similar to the ones described in chapter 4 in the case of a full-mesh and hybrid management substrate structure.

With the objective to offload traffic from as many utilised line cards as possible, one of the key decisions in the proposed approach concerns the bundled link to consider for (a) removing traffic from, and (b) assigning that traffic to, at each iteration of the reconfiguration process. In order to be turned off, the LCs need to be non-utilised. This implies that they must be entirely offloaded. Intuitively, it is easier to entirely offload lightly utilised LCs as there may be a larger number of alternative solutions to accommodate the diverted traffic. Therefore, the proposed approach aims at diverting traffic away from the LCs with low utilisation towards the LCs with high utilisation. The decision regarding traffic diversion is based on a list of all utilised LCs in the network, ranked according to their load ($\rho_{\text{LCU}}(l)$)[1], which is common to all substrate nodes. Ranking is based on the load of utilised LCs, or on the collective load of BLs with a utilised LC, in increasing order (i.e. from the lowest to the highest utilisation). Based on the load $\rho(l)$ and the LC capacity $c_{\text{LC}}(l)$ of a bundled link $l$, $\rho_{\text{LCU}}(l)$ can be determined as follows[2]:

$$\rho_{\text{LCU}}(l) = c_{\text{LC}}(l) \cdot \big(\rho(l) - \lfloor \frac{\rho(l)}{c_{\text{LC}}(l)} \rfloor \cdot c_{\text{LC}}(l)\big) \tag{5.1}$$

The ranked list can be computed and provided to all substrate nodes by a distributed monitoring mechanism such as the one proposed in [83]. The goal of the DE is to move the traffic load of the first LC in the list to another LC further down the list that can accommodate this load and thus potentially fill-up its remaining capacity. This involves the computation of new splitting ratios by the DE, which results in different traffic volumes sent over each of the vir-

[1] This represents the load on the utilised line card in bundle $l$.

[2] $\big(\rho(l) - \lfloor \frac{\rho(l)}{c_{\text{LC}}(l)} \rfloor \cdot c_{\text{LC}}(l)\big)$ is the remainder of the division of $\rho(l)$ by $c_{\text{LC}}(l)$ and its value is between 0 and 1.

tualised MTR planes. The list is re-ranked at each iteration and, with the principle that traffic cannot be assigned to a LC from which traffic has been removed in a previous iteration, the process terminates when all the LCs from which traffic can be removed have been exhausted. To prevent the disconnection of source-destination pairs during the reconfiguration process, the approach does not allow a critical BL (one that is used by a source-destination pair in every virtual topology) to be assigned with a zero splitting ratio, i.e. at least one LC is active (either *full* or *utilised*) in that BL. The algorithm is detailed in the next section.

## 5.4 Adaptive Traffic Distribution Algorithm

Previous work in energy-aware networking identified that the optimal allocation of traffic to achieve green objectives is an NP-hard problem [127][132]. To satisfy the strict time constraints of an online reconfiguration process, such as the one proposed, an efficient heuristic algorithm was devised. The main functionality of this algorithm involves determining the BLs to remove traffic from and re-assign that traffic to, as well as the amount of traffic to shift.

### 5.4.1 Definitions

Each bundled link is associated with a state $S$, which is expressed with two bits as follows:

- $S_{00}$ - traffic can be both removed from and assigned to a BL.
- $S_{01}$ - traffic can be removed from, but cannot be assigned to a BL.
- $S_{10}$ - traffic cannot be removed from, but can be assigned to a BL.
- $S_{11}$ - traffic can neither be removed from, nor can be assigned to a BL.

When generating a new configuration, in addition to the traffic load on the utilised LCs of BLs, the algorithm requires the amount of spare capacity ($s(l)$) on utilised LCs for these links. This is determined as follows:

$$s_{\text{LCU}}(l) = c_{\text{LC}}(l) - \rho_{\text{LCU}}(l) \tag{5.2}$$

### 5.4.2 Traffic Re-assignment - Where?

At each iteration, the reconfiguration entity (i.e. DE) executing the algorithm determines the volume of traffic that can be re-directed to offload a utilised LC. This is achieved by selecting the first element from the ranked list of utilised LCs and determining all the traffic flows emanating from the reconfiguration entity that are routed over the associated BL. Without loss of generality, let $LCU_0$ represent the first LC in the list and $l_0$ the corresponding BL. Any flows for which $l_0$ is a critical link are disregarded so that the physical topology remains connected.

The next step is to determine, for each of these flows, the virtual topologies from which traffic can be removed and the ones to which this traffic can be re-assigned. This is signified by the two state bits of $l_0$ as defined in the previous section, which are updated at each iteration. To achieve the energy reduction objective, the offloading of full LCs or the traffic assignment to non-utilised LCs is not allowed. Although this may remove flexibility from the algorithm towards obtaining an optimal solution, this reduces the complexity of the problem and facilitates the computation of a new feasible assignment of traffic. BLs having all their LCs either full or non-utilised are set to state $S_{11}$. BLs with a utilised LC are set to state $S_{00}$. The flows over a virtual topology using $l_0$ are marked for removal if all the BLs along their paths have a first state bit of $0$. Conversely, the alternative topologies (where $l_0$ is not used) over which traffic can be re-directed are the ones that have all the BLs with a second state bit of $0$ along the paths of the considered flows. This results in a list of flows that can be removed from $LCU_0$ and assigned to other utilised LCs.

### 5.4.3 Traffic Re-assignment - How Much?

Once the flows that can potentially be shifted have been determined, they are removed iteratively from the relevant topologies. This process terminates if the list of flows is exhausted or if the entire load on $LCU_0$ has been removed. It may not always be possible to move the entire volume of a flow since the available capacity on alternative virtual topology paths acts as a constraint. For this reason, the proposed algorithm computes the maximum volume that can be removed from each flow in three steps:

1. For each topology where $LCU_0$ is involved, the LC with the least load along the associated path is determined. The minimum of these values over the relevant topologies is subsequently computed, $\rho^{\text{min}}_{\text{LCU}}$.

2. For each alternative topology where $LCU_0$ is not involved, the utilised LC with the least spare capacity along the associated path is determined. The minimum of these values over the relevant topologies is subsequently computed, $s^{\text{min}}_{\text{LCU}}$.

3. The maximum volume of traffic to remove from $LCU_0$ is

$$\begin{cases} s^{\text{min}}_{\text{LCU}} & \text{if } \rho^{\text{min}}_{\text{LCU}} > s^{\text{min}}_{\text{LCU}} \\ \rho^{\text{min}}_{\text{LCU}} & \text{otherwise} \end{cases} \tag{5.3}$$

   In the case where $LCU_0$ is involved in multiple virtual topologies, traffic is removed equally between them. It should be noted that while this constraint may prevent the algorithm from obtaining an optimal solution, it reduces of the complexity of the approach.

Having determined the volume of traffic to remove from each topology, the algorithm ranks the utilised LCs with the least spare capacity, computed in step 2) above, in decreasing order. Starting from the top of this list, the removed traffic is assigned to the relevant topology, filling up the utilised LCs, until the entire traffic volume has been allocated. The resulting total traffic volume assigned to each virtual topology enables the algorithm to compute new splitting ratios.

At each iteration of the reconfiguration process, the ranked list of utilised LCs is updated and new BL states are set. State $S_{11}$ is assigned to a BL whose LCs become either full or non-utilised and state $S_{01}$ to a BL if $LCU_0$ was a constituent LC, but its entire load was not successfully removed in the previous iteration. The latter state is communicated to all nodes of the management substrate so that traffic is not assigned to the associated utilised LC during the next iteration. Alternate actions of traffic removal and assignment on the same LC (flip-flops), which can lead to configuration instabilities, can thus be avoided.

## 5.5 Evaluation

To evaluate the performance of the proposed energy-aware approach, real topologies and traffic traces from the two PoP-level networks Abilene [99] and GEANT [100] were used. In order to perform the experiments, the Java-based software developed for the load balancing application (chapter 4 section 4.6.1) was extended with the implementation of the energy-saving algorithm. The software was, in particular, adapted to support the link aggregation model considered in this chapter. This section presents the results from the experiments in terms of LC gain and maximum utilisation. The experiments were performed on a laptop with a 2.80 GHz Intel Core i7-2640M processor and 8 GB memory.

### 5.5.1 Experimentation Setup

672 traffic matrices (TMs) were used for all experiments. These cover a period of 7 days so that a wide range of traffic conditions are taken into account. Four virtual planes were used for both Abilene and GEANT, which were generated according to the algorithm described in chapter 4 section 4.3.3. With the bundled LCs having the same capacity, Table (5.3) summarises the configuration of the BLs in the two physical topologies. The 28 unidirectional BLs of Abilene are implemented with 112 LCs and the 68 BLs of GEANT with 210 LCs.

The performance of the proposed approach (abbreviated NRG on plots and tables) is compared against:

- basic MTR (MTR_basic) with static splitting ratios (do not adaptively change) equal to the inverse of the capacity of the bottleneck bundled link in each virtual topology.

| | Abilene | | GEANT | | |
|---|---|---|---|---|---|
| | **Type 1** | **Type 2** | **Type 1** | **Type 2** | **Type 3** |
| **BL capacity (Gbps)** | 10 | 2 | 10 | 2 | 1 |
| **BL size** | 4 | 4 | 4 | 2 | 2 |
| **LC capacity (Gbps)** | 2.5 | 0.5 | 2.5 | 1 | 0.5 |
| **Number of BLs** | 26 | 2 | 37 | 15 | 16 |
| **Number of LCs** | 104 | 8 | 148 | 30 | 32 |

Table 5.3: Abilene and GEANT bundled links (BLs) configuration.

- the adaptive resource management scheme (LB) described in chapter 4 with changing splitting ratios that achieve a load balancing objective.

Adaptation is performed every 15 minutes in both Abilene and GEANT topologies with a maximum of 50 algorithm iterations. The splitting ratios generated in the previous cycle are initially used at the start of each reconfiguration cycle. For each of the schemes, the gain (G) in terms of number of LCs used to accommodate the traffic is measured, as well as the maximum utilisation in the network (max-u), i.e. typically used as a measure of the load balancing level. In addition to evaluating the proposed approach under the load imposed by the available TMs, its performance is investigated under heavily loaded traffic conditions. Instead of scaling all the TMs, which increases the size of every flow, background traffic that fully consumes one LC on every BL is introduced. This selectively increases the size of only some flows. The results presented in this section are structured according to the presence and absence of background traffic.

### 5.5.2 Definitions

The following variables, which are related to the BLs in the network, are used when evaluating the performance of the proposed approach: the bundle size ($n(l)$), i.e. the number of LCs that implement a BL, the number of full ($n_{\text{LCF}}(l)$), utilised ($n_{\text{LCU}}(l)$), and non-utilised ($n_{\text{LCN}}(l)$) LCs. These are calculated as follows:

$$n(l) = \frac{c(l)}{c_{\text{BL}}(l)} \tag{5.4}$$

$$n_{\text{LCF}}(l) = \lfloor \frac{\rho(l)}{c_{\text{BL}}(l)} \rfloor \tag{5.5}$$

$$n_{\text{LCU}}(l) = \begin{cases} 0 & \text{if } \big(\rho(l) - \lfloor \frac{\rho(l)}{c_{\text{LC}}(l)} \rfloor \cdot c_{\text{LC}}(l)\big) = 0 \\ 1 & \text{otherwise} \end{cases} \tag{5.6}$$

$$n_{\text{LCN}}(l) = n(l) - \big(n_{\text{LCF}}(l) + n_{\text{LCU}}(l)\big) \tag{5.7}$$

| Gain (%) | Abilene (%) | | GEANT (%) | |
|---|---|---|---|---|
| | NRG vs. MTR_basic | NRG vs. LB | NRG vs. MTR_basic | NRG vs. LB |
| **Minimum** | 6.06 | 0 | 17.02 | 17.02 |
| **Maximum** | 32.43 | 34.21 | 51.06 | 51.06 |
| **Average** | 21.85 | 18.03 | 46.10 | 46.08 |

Table 5.4: Percentage gain with no background traffic in the case of ranking based on line card load.

The gain G associated with an energy-aware reconfiguration (NRG) compared to a non-NRG one, e.g. basic MTR or LB, is defined as follows:

$$G = 1 - \frac{(N_{\text{LCF}} + N_{\text{LCU}})_{\text{NRG}}}{(N_{\text{LCF}} + N_{\text{LCU}})_{\text{nonNRG}}} \tag{5.8}$$

where the total number of full ($N_{\text{LCF}}$) and utilised ($N_{\text{LCU}}$) LCs in the network are given by:

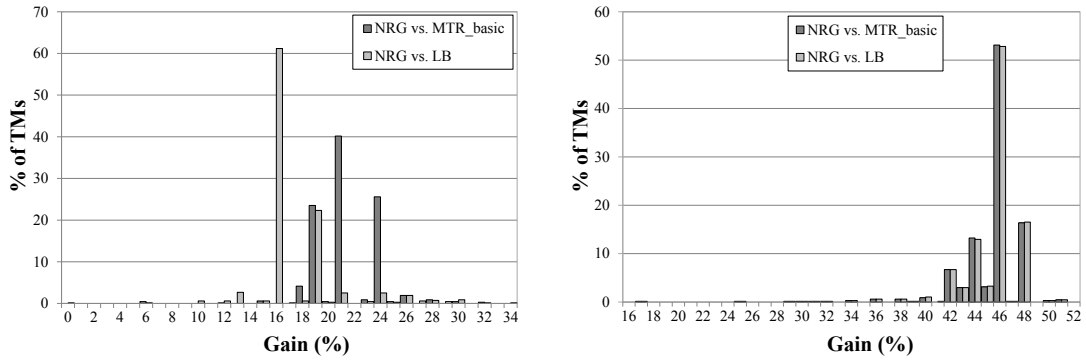
$$N_{\text{LCF}} = \sum_{l \in \mathcal{L}} n_{\text{LCF}}(l) \tag{5.9}$$

$$N_{\text{LCU}} = \sum_{l \in \mathcal{L}} n_{\text{LCU}}(l) \tag{5.10}$$

### 5.5.3 Algorithm Performance - No Background Traffic

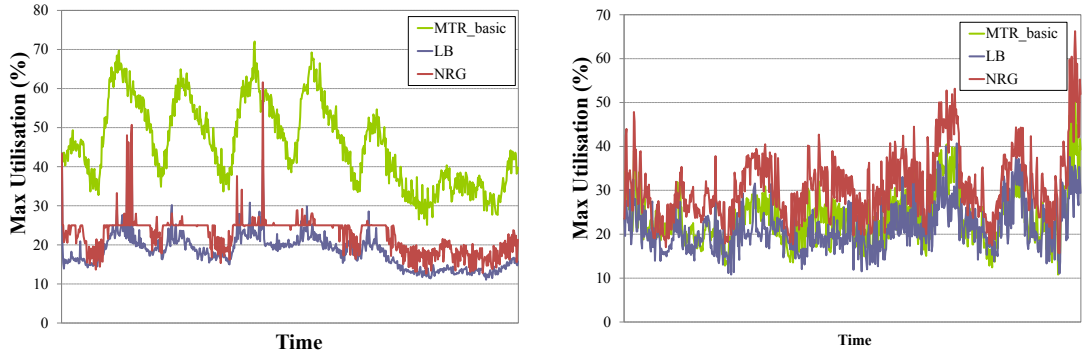
The first set of experiments concerns the case where background traffic is not present. Table (5.4) summarises the performance of the proposed approach (NRG), in terms of active LCs, compared to MTR_basic and LB for the Abilene and GEANT topologies, in the case utilised LCs are ranked according to their load in increasing order.

A much higher average gain can be observed in the case of GEANT compared to Abilene. The NRG approach performs better on average compared to MTR_basic (22%) rather than LB (18%) in the case of Abilene, but the performance is very similar in the case of GEANT. There were no instances where the gain is negative. Figure (5.2a) plots the distribution of the gain for the Abilene network. Compared to MTR_basic, the NRG approach achieves a gain between 19% and 24% for more than 85% of the TMs, with the most frequent gain of 21% applying to around 40% of the TMs. The comparison with LB shows that a gain between 16% and 19% is achieved for more than 85% of the TMs, with the most frequent gain of 16% applying to around 60% of the TMs. In both comparisons, the gain achieved by the NRG approach is always non-negative, with the worst case being the configuration for only one TM, for which there was no gain. Compared to LB, the gain is slightly smaller on average than that of MTR_basic, but




(a) Abilene (b) GEANT

Figure 5.2: Gain distribution for the Abilene and GEANT networks.




(a) Abilene (b) GEANT

Figure 5.3: Evolution of the maximum utilisation in the Abilene and GEANT networks.

comparable in terms of order of magnitude. The results for the GEANT topology in figure (5.2b) show that the NRG approach achieves a gain between 44% and 48% compared to both MTR_basic and LB for more than 85% of the TMs, with the most frequent gain of 46% applying to around 53% of the TMs. As in the case of Abilene, the gain is always non-negative.

Figures (5.3a) and (5.3b) plot the evolution of the maximum utilisation (max-u) under the three different schemes in the Abilene and GEANT networks, respectively. For Abilene, the maximum utilisation achieved by the NRG approach is significantly lower than that of MTR_basic, but higher than that of LB by 25% on average. Flat-lining for NRG occurs because the BL with the highest utilisation remains the same over a period of time. In the case of GEANT, max-u for NRG follows the same trend but is higher than that of the two other schemes, with an average difference of 47% compared to LB - almost double the deviation compared to Abilene.

| Gain (%) | Abilene (%) | | GEANT (%) | |
|---|---|---|---|---|
| | NRG vs. MTR_basic | NRG vs. LB | NRG vs. MTR_basic | NRG vs. LB |
| **Minimum** | -12.12 | -19.35 | 17.02 | 17.02 |
| **Maximum** | 28.10 | 23.33 | 52.17 | 52.17 |
| **Average** | 16.38 | 13.93 | 46.70 | 46.70 |

Table 5.5: Percentage gain with no background traffic in the case of ranking based on bundled link load.

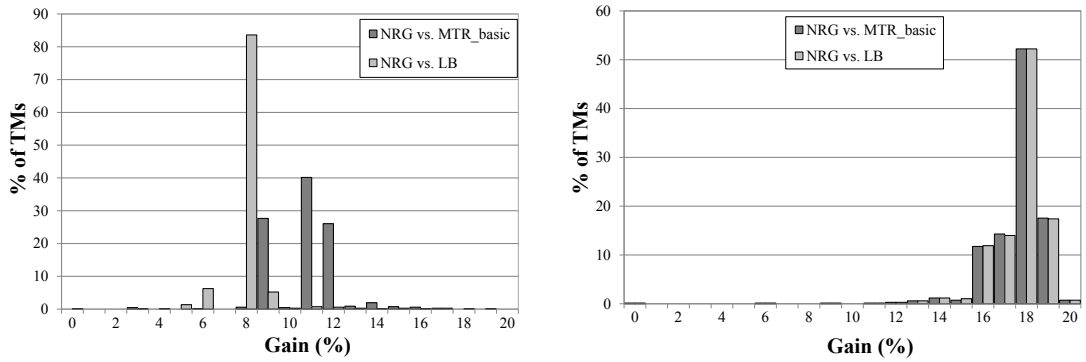
| Gain (%) | Abilene (%) | | GEANT (%) | |
|---|---|---|---|---|
| | NRG vs. MTR_basic | NRG vs. LB | NRG vs. MTR_basic | NRG vs. LB |
| **Minimum** | 3.17 | 0 | 6.61 | 6.61 |
| **Maximum** | 17.91 | 19.12 | 20.33 | 20.32 |
| **Average** | 11.33 | 8.41 | 17.92 | 17.91 |

Table 5.6: Percentage gain with background traffic in the case of ranking based on line card load.

The performance obtained in terms of energy savings in the case where the ranking is based on the collective load of BLs with a utilised LC are also evaluated. Results are presented in Table (5.5). A significant degradation in terms of performance can be observed in the case of Abilene. The NRG scheme can even be outperformed by the two other approaches in few cases. The objective of the energy management algorithm is to shift traffic away from as many line cards as possible so that these can be turned off. Due to the greedy nature of the algorithm, the performance is affected by the initial configuration state. Intuitively, a better gain in terms of energy savings is expected when traffic is iteratively removed from the least loaded line cards. It is more likely that the entire traffic load of the lightly loaded line cards can be accommodated elsewhere, and as such, that these can be switched off. In contrast, considering a ranking based on the BL load may provide less flexibility for removing the entire traffic load from some of the line cards. It can be noted that similar performance can be achieved in the case of GEANT, which indicates the existence of an overlap between the lists of line cards obtained according to the two ranking methods.

### 5.5.4 Algorithm Performance - Background Traffic

Table (5.6) summarises the performance of the NRG approach, in the presence of background traffic, compared to MTR_basic and LB for the Abilene and GEANT topologies in the case utilised LCs are ranked according to their load in increasing order.



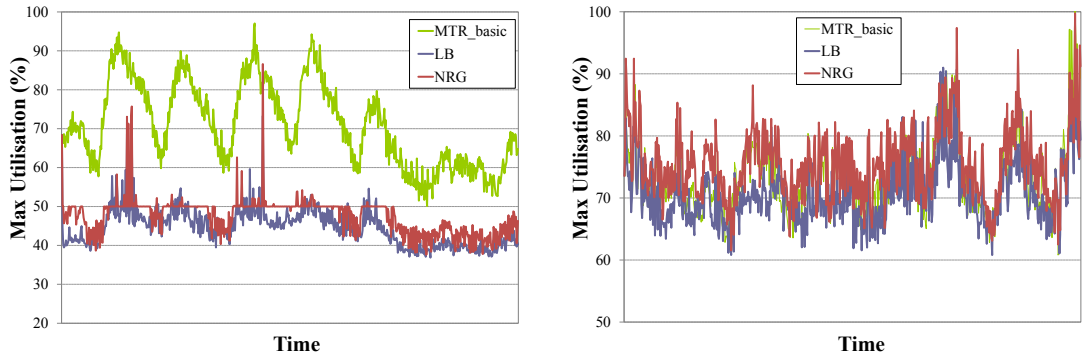

(a) Abilene (b) GEANT

Figure 5.4: Gain distribution for the Abilene and GEANT networks.

As in the absence of background traffic, a higher average gain can be observed in the case of GEANT compared to Abilene. The NRG approach performs better on average compared to MTR_basic (11%) rather than LB (8%) in the case of Abilene, but the performance is very similar in the case of GEANT. There were no instances where the gain is negative. Figure (5.4a) plots the distribution of the gain for the Abilene network. Compared to MTR_basic, the NRG approach achieves a gain between 9% and 12% for more than 90% of the TMs, with the most frequent gain of 11% applying to around 40% of the TMs. The comparison with LB shows that a gain between 6% and 9% is achieved for more than 90% of the TMs, with the most frequent gain of 8% applying to around 84% of the TMs. Compared to LB, the gain is slightly smaller on average than that of MTR_basic, but comparable in terms of order of magnitude. The results for the GEANT topology in figure (5.4b) show that the NRG approach achieves a gain between 16% and 19% compared to both MTR_basic and LB for more than 90% of the TMs, with the most frequent gain of 18% applying to around 52% of the TMs. As in the case of Abilene, the gain is always non-negative.

Figures (5.5a) and (5.5b) plot the evolution of max-u under the three different schemes in the Abilene and GEANT networks, respectively. For Abilene, the maximum utilisation achieved by the NRG approach is significantly lower than that of MTR_basic, but higher than that of LB by 7% on average. In the case of GEANT, max-u for NRG is higher than that of the two other schemes, but not as much as in the case of Abilene, with an average deviation of 7% compared to LB.

Finally, the performance obtained in terms of energy savings in the case where the ranking is based on the collective load of BLs with a utilised LC are presented in Table (5.7). As in the case of no background traffic, a performance degradation can be observed in the case of




(a) Abilene (b) GEANT

Figure 5.5: Evolution of the maximum utilisation in the Abilene and GEANT networks.

| Gain (%) | Abilene (%) | | GEANT (%) | |
|---|---|---|---|---|
| | NRG vs. MTR_basic | NRG vs. LB | NRG vs. MTR_basic | NRG vs. LB |
| **Minimum** | 0 | -1.60 | 6.61 | 6.61 |
| **Maximum** | 14.06 | 16.17 | 19.71 | 19.71 |
| **Average** | 10.03 | 7.36 | 17.56 | 17.56 |

Table 5.7: Percentage gain with background traffic in the case of ranking based on bundled link load.

Abilene while similar performance is obtained in the case of GEANT.

### 5.5.5 Analyis and Discussion

The gain distribution plots illustrate strong concentration around the average in all cases indicating that the performance of the proposed approach is consistent with most TMs. The higher gain observed in the case of GEANT compared to Abilene, in both the presence and absence of background traffic, is attributed to the variation in the GEANT BL capacities. Offloading LCs of smaller capacity is a simpler task since less traffic needs to be shifted. As such, the capacity homogeneity of the Abilene BLs (only 6.6% are 2 Gb/s links) acts as a limiting factor. Furthermore, it is evident that the average gain in the presence of background traffic, for both GEANT and Abilene networks, is lower. This is because the proportion of utilised LCs (that can be switched off) in the total number of active LCs (sum of utilised and full) is always lower under loaded traffic conditions. As expected, the lowest max-u in all experiments is obtained by LB, which is a load balancing approach with the objective of minimising the maximum utilisation in the network. The difference in max-u obtained by NRG for Abilene and GEANT is generally insignificant, demonstrating that energy conservation may come with a degree of load balancing.

In a similar fashion to the load balancing application presented in chapter 4, the performance of the energy management heuristic depends on the characteristics of the paths obtained through the computation of the virtual routing planes, and, as such, is driven by the characteristics of the underlying physical topology. In addition, the settings of the LCs can also affect the performance. In general, the choice of the number and capacity of the LCs in each bundled link directly influences the flexibility of the algorithm when determining how to re-allocate the traffic load. In particular, the use of a large number of small capacity LCs in each BL may lead to better performance than the use of fewer big capacity LCs.

Given that the proposed algorithm is executed by ingress nodes, its time complexity is theoretically defined by the number of local traffic flows. This is similar to the time complexity of the load balancing algorithm (chapter 4), i.e. in the order of $O(N^2)$ in the case a PoP-level topology with $N$ nodes. In practice, however, the complexity is directly driven by the actual number of flows that need to be considered such that all the traffic can be removed from a utilised LC. In the best case, a solution can be determined by considering a single local flow. The ratio of the average execution time of the NRG algorithm to the total number of local traffic flows is 0.13 for Abilene and 0.03 for GEANT. This suggests that, despite the fact that GEANT has almost four times as many local flows as Abilene (420 compared to 110), the execution time for the two topologies is similar. This result shows that the time complexity does not increase quadratically in practice since a similar number of flows are considered for determining a solution in both Abilene and GEANT. The complexity of the overall adaptation process is influenced by the number of reconfiguration iterations since this incurs communication overhead, as explained in chapter 4.

## 5.6 Conclusion

In this chapter, a new resource management approach for IP networks that addresses the very important issue of energy efficiency is presented. The objective is achieved by controlling the traffic distribution through an intelligent substrate at the edges of the network. In contrast to the majority of previous work that suggests switching off entire links, the switching off strategy is applied at the line card level. By performing capacity adaptation instead of topology adaptation, the proposed approach is more flexible since it prevents the disconnection of source-destination pairs during the reconfiguration process. In conjunction with multi-topology routing, this allows the computation of alternative paths in advance, thus reducing complexity at the routing level. Furthermore, dynamic traffic conditions are considered at short or long scales. This allows the proposed approach to adapt to the dynamicity of fixed infrastructures, following changes in user

generated traffic load and available network applications. Of course, this level of complexity could be tackled due to the previous reasonable assumptions. Monitoring of dynamic conditions is a complicated subject that requires an independent infrastructure. In this work, it is assumed that the information management overlay proposed in [83] is used. The decentralised and online nature of the approach avoids the scalability problems encountered by centralised proposals. In addition, the efficiency of the heuristic employed by the proposed scheme makes it suitable for an online reconfiguration process compared to the rather 'heavy' algorithms previously proposed. Last, but not least, an important aspect of the approach is deployability, since the logic to realise the energy objective requires modifications only to the ingress nodes of a network, but not to the core nodes.

# Chapter 6

# In-network Cache Management

## 6.1 Introduction

Content Delivery Networks (CDNs) have been the prevalent method for the efficient delivery of rich content across the Internet. In order to meet the growing demand for content, CDN providers deploy massively distributed storage infrastructures that host content copies of contracting content providers and maintain business relationships with Internet Service Providers (ISPs). Surrogate servers are strategically placed and connected to ISP network edges [140] so that content can be closer to clients, thus reducing both access latency and the consumption of network bandwidth for content delivery.

Current content delivery services operated by large CDN providers like Akamai [141] and Limelight [142] can exert enormous strain on ISP networks [78]. This is mainly attributed to the fact that CDN providers control both the placement of content items in surrogate servers spanning different geographic locations, as well as the decision on where to serve client requests (i.e. server selection) [143]. These decisions are taken without knowledge of the precise network topology and state in terms of traffic load and may result in network performance degradation.

In this chapter, a cache management approach with which ISPs can have more control over their network resources is proposed. Exploiting the decreasing cost of storage modules, the proposed approach involves operating a limited capacity CDN service within ISP networks by deploying caches at the network edges (figure (6.1)). These can be external storage modules attached to routers or, with the advent of flash drive technology, integrated within routers. Such a service can cache popular content items, specific to an ISP, and serve most client requests from within the network instead of fetching content items from surrogate/origin servers. Empowering ISPs with caching capabilities can allow them to implement their own content placement and server selection strategies which will result in better utilisation of network resources [9]. In addition, there are economic incentives for an ISP to adopt this approach given that traffic on

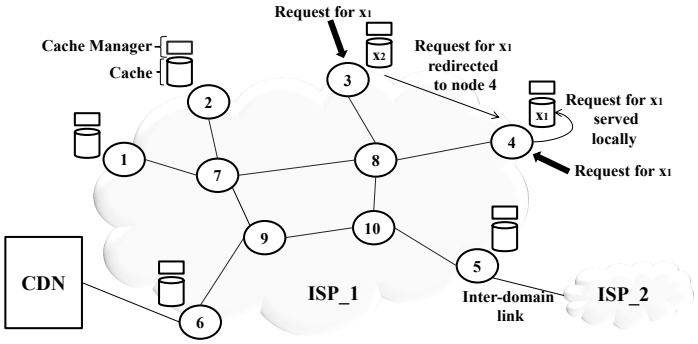



Figure 6.1: Overview of the proposed caching infrastructure.

inter-domain links can decrease significantly. In order to deploy such an approach, new interaction models between ISP and CDN providers may need to be defined. These would address issues relating to the resolution of content requests and the exchange of necessary information, e.g. content items to cache, their popularity and size. An alternative solution for obtaining content item popularity/size information would involve the ISP maintaining records of previous user requests from which it can infer future demand. This prediction could be enhanced with information from other sources like other ISPs as well as CDNI [144]. Although these are challenging issues, the focus of this work was on resource management mechanisms. These other issues are to be addressed in future extensions of this Ph.D. work.

While the utilisation of network resources is affected by both content placement and server selection operations, this work concentrates on the former. Research CDNs, such as Coral [145], have proposed distributed management approaches [146]. However, commercial CDNs have traditionally used centralised models for managing the placement of content items in distributed surrogate servers. Complex algorithms are executed in an offline fashion for determining the optimal placement of content item copies for the next configuration period (typically in the order of days). With the objective of keeping the cost and the complexity of the approach low, simple strategies are employed in this study to decide on the placement of content item copies in the various caching points. Such strategies do not incur significant processing and communication overhead among distributed decision points and their functionality can thus be realised by commodity hardware components. In contrast to the ISP-centric caching approaches proposed in [147] and [148], which exclude CDNs from the delivery chain, the solution proposed in this chapter maintains the CDN presence but only for the purpose of providing content which is not locally cached by an ISP. This will ensure content item availability given the global footprint of large CDN providers.

A distributed in-network cache management approach has also been proposed in [80],

which focuses on content *replacement* strategies operated at short timescales. The approach proposed here is more geared towards longer reconfiguration periods (in the order of hours) and can be complemented by online solutions that optimise the inital content placement configuration in the face of dynamic changes in user demand. Two categories of strategies that can be used by an ISP to control the placement of content items at the different network caches are described in this chapter. In the first category, only information about the demand from locally connected users is used to determine the configuration of each of the caches, whereas, in the second, knowledge about global content item popularity and interests in the different caching locations is required. In order to analyse how the proposed strategies can be affected by user demand, a methodology was developed and specific metrics were defined to characterise the user demand profile. The performance of the proposed strategies, in terms of network resource utilisation, was evaluated based on a wide range of user demand profiles, and the obtained performance was compared according to proposed metrics.

The rest of the chapter is organised as follows. Section 6.2 discusses related work. Section 6.3 provides an overview of the proposed approach and introduces the main modeling assumptions of this work. Section 6.4 presents the details of two categories of content placement strategies. Section 6.5 describes the methodology used to evaluate the performance of the proposed strategies and analyses the evaluation results. A summary of the work and future directions is finally presented in section 6.6.

## 6.2 Related Work

The placement problem has received a lot of attention from the research community over the years and has been addressed in different contexts. For instance [149] investigates the problem for the selection of the most appropriate physical locations to place web server replicas in order to minimise network cost.

Closer to the work presented in this chapter are the approaches proposed in [150] [151] [152] and [80], which all focused on intelligent techniques to replicate content items across different network locations in order to better utilise network resources. Previous work, such as [153], has also investigated specific solutions for hierarchical network infrastructures, which have been applied in the context of IPTV [154][152].

A distributed placement strategy in the context of distributed replication groups is proposed in [151], where nodes in the group cooperate to take replication decisions that can minimise the overall network cost. In that work, the authors assume the existence of a constant and uniform cost for retrieving any content item within the group, and, as such, they do not take into account

network characteristics such as link utilisation. The content placement problem has also been tackled from the point of view of game theory in [155] under the conditions of infinite cache capacity. An autonomic cache management framework for information-centric networks was presented in [80]. Three dynamic distributed replacement algorithms with different levels of cooperation between the caches and complexity were evaluated. These could complement the approach proposed in this chapter by optimising the initial content placement configuration according to the short-term dynamics of user demand.

Optimal solution structures for the combined problem of object placement and assignment of requests to caching locations (i.e. server selection) were investigated in [156], [157] and [79], where the objective was to minimise the global network cost. Although the proposed algorithms can provide near-optimal solutions to the coupled optimisation problem (i.e. content placement and server selection), the approaches have limitations in practice. They assume, in particular, the availability of global information about user demand and network conditions at a centralised location, which may incur a significant overhead and can have scalability limitations. In addition, given that the two problems are solved concurrently, the solution can take a long time to converge.

Given the significant impact that content delivery has on the utilisation of ISP networks, some work has recently started to investigate new models and frameworks to support the interaction between ISPs and CDNs [78][143][147][148]. In [78], the authors highlight that CDN providers and ISPs can indirectly influence each other, by performing server selection and traffic engineering operations respectively, and they investigate different models of cooperation between the two entities. In [143], the authors propose a framework to support joint decisions between a CDN and an ISP with respect to the server selection process. This framework allows the ISP and the CDN to collaborate by exchanging some local information (network utilisation from the ISP side and server conditions from the CDN side), so that it can result in better control of the resources. In contrast to these approaches, the proposed solution focuses on empowering ISPs with caching capabilities, which can allow them to implement their own content placement and server selection strategies. ISP-centric caching approaches have also been considered in [147] and [148]. However, the full-blown CDN service to be supported by an ISP, as proposed in these approaches, can incur high operational costs, given that ISPs will have to maintain large storage capacities, and may thus be an economically unviable solution.

## 6.3 Content Distribution Management

### 6.3.1 Problem Statement

The scenario considered in this work is depicted in figure (6.1). A set of caches are deployed at the network edges, so that a cache is associated with every network edge node. In addition, each cache is locally controlled by a cache manager which is responsible for enforcing caching decisions.

Unlike the heavyweight caching infrastructure maintained by CDN providers, the proposed solution allows the ISP to operate a simpler and more lightweight caching service. In this case, the total caching space available in the network may not permit in practice the storage of all possible content items provided by the CDN. A subset of the content items that the ISP wishes to cache in its network (for instance the most popular ones) needs therefore to be pre-selected, so that the volume of selected content items does not exceed the total caching capacity. In this work, it is assumed that the list of content items can be decided in a collaborative manner between the ISP and the CDN according to pre-established business agreements between the two stakeholders. As such, information about the content items (e.g. popularity, size) could be available to the ISP, which could be provided by the CDN or directly inferred by the ISP. In this context, the selected items are those for which the popularity characteristics can be estimated based on monitored user request statistics and/or based on external sources of information such as the website front page of the relevant services (e.g. BBC iPlayer [158]).

Since the total content volume that can be stored in each cache depends on the cache capacity, it may not be possible to accommodate all the content items at each individual caching location. In this case, the content items are distributed across the different caching locations, so that each content item is stored in at least one cache in the network. As a result, user requests for content item can be served from within the network instead of being redirected to CDN servers. As depicted in figure (6.1), if a request for content item $x_1$ is received at edge node $4$ and content item $x_1$ is available in the local cache, it is served locally. Otherwise, the request is redirected to one of the caches in the network where $x_1$ is stored. For instance, a request for content item $x_1$ received at node 3 is redirected to and served by node $4$.

Serving a content request from the same location as the request directly from the access node (i.e. locally) does not affect the utilisation of network resources. Retrieving a content item from a remote caching location, however, generates network traffic. Intelligently managing the placement of the content in the different caches, under specific user demand characteristics, can therefore allow an ISP to control the utilisation of network resources.

### 6.3.2 Intelligent Content Placement Approach

Given a set of $M$ caches $m$ with capacity $c_{\mathrm{m}}$, and a list of $X$ content items $x$ with size $s_{\mathrm{x}}$, controlling the placement of content consists in determining the configuration of each of the caches, according to the current user demand, so that the network resources can be used more effectively. More specifically, the placement problem is to determine the number of copies of each content item to store in the network, as well as the location of each copy, so that the two following constraints are satisfied: 1) the placement of content in each individual cache satisfies the cache capacity constraint, and, 2) each content item is cached in at least one caching location. The resulting content placement is then used as an input by a server selection algorithm which determines how to allocate user requests to the caches. Although simple algorithms can be used in practice (e.g. a round-robin mechanism), linear programming is used to formulate the problem and compute the optimal request allocations. This allows to focus on the content placement heuristics.

Several traditional traffic engineering metrics can be used to optimise the utilisation of network resources. In this work the objective is to minimise the maximum link utilisation (max-u), which is commonly used in the traffic engineering literature [159][65]. As shown in previous work, determining the optimal placement of content items that minimises the max-u in the network is an NP-hard problem [79]. This work investigates practical heuristics that can be used to solve this problem.

Resource management approaches usually rely on centralised solutions executed by a manager that has a global view of the current conditions (e.g. user demand, average link utilisation). Although relatively simple to implement, centralised solutions have limitations in practice. In addition to the single point of failure issue, collecting global information about user demand and network conditions poses scalability problems since they can incur significant traffic overhead.

This work investigates an approach where, instead of relying on placement decisions received from a centralised manager, distributed cache managers coordinate among themselves to decide how to efficiently use the available caching space with the objective of better utilising the network resources. To achieve this objective, cache managers are organised into an intelligent in-network management substrate similar to the one described in chapter 3 and used for the purpose of adaptive resource management in chapters 4 and 5. The substrate is a logical structure used to facilitate the communication between the cache managers, which enables them to decide on reconfiguration actions in a coordinated manner. In traditional CDN infrastructures, the common practice is to perform reconfigurations (i.e. cache updates) periodically [160]. Previous research showed that, for video content items, content placement updates should be applied

at a timescale in the order of hours in order to minimise the impact on the network and users [161][79]. In this chapter, it is assumed that the frequency of content placement decisions is in the order of hours.

### 6.3.3 Modeling Choices

Several factors can affect the performance of a placement strategy, such as the user demand and the cache size. Selecting the strategy to apply is therefore a challenging issue. This work focused on the influence of user demand characteristics on the performance of the proposed placement strategies. In order to limit the influence of other factors, the following modeling choices and assumptions were considered.

The total volume of content items to cache in the network, $V_{\mathrm{X}}$, is defined as the sum of the size of each content item $x$, so that $V_{\mathrm{X}} = \sum_x s_{\mathrm{x}}$. In addition, the total caching space available in the network, $V_{\mathrm{M}}$, is defined as the sum of the capacity of each cache $m$, so that $V_{\mathrm{M}} = \sum_m c_{\mathrm{m}}$. By design, $V_{\mathrm{M}}$ is chosen so that each content item can be stored in at least one of the network caches, i.e. $V_{\mathrm{X}} \leq V_{\mathrm{M}}$. This ensures that the choice of the content items to cache in the network is a constant of the problem, and as such, enables the direct comparison of the outcome of each placement strategy (i.e. replication degree and location of each content item). In addition, in order to uncorrelate the influence of the size of the content items, it is assumed that they all have the same size. In this case, the storage cost associated with the placement of a content item at a specific location is independent of the size of the item, and as such, can be considered as a constant. In the case of items with different sizes, however, the storage cost is a function of the content items, which can influence the decision logic of a particular placement strategy, and subsequently affect the performance. It is also assumed that all caches have the same capacity. The size of a cache at a particular location can depend for instance on several topological factors (e.g. node centrality). Determining the correlation between content item size and user interest for the content item, as well as taking into account different cache sizes, are challenging research issues but are outside the scope of this work in which the problem formulation is simplified in order to limit the influence of other factors. The user demand is characterised by a) the total volume of requests for each content item in the network (i.e. global content popularity - GCP), and b) the number of caching locations where each content item is requested (i.e. geographic distribution of the interests - GDI).

**Modeling the global popularity of the contents** Given that content popularity is long-tail distributed, it is commonly accepted that the GCP distribution can be represented by the Zipf law [162]. In this work, a Zipf distribution of parameter $\alpha$ (scaling-law coefficient) is used to model the GCP. Content items are indexed according to their rank $r$ in the distribution, so that the

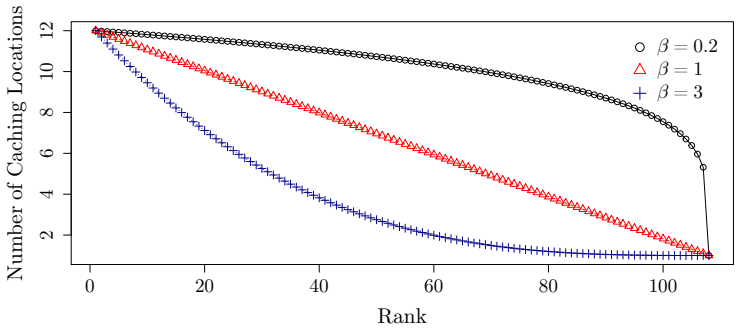



Figure 6.2: Profile of the function $f_\beta$ for three values of $\beta$.

global popularity of a content item decreases when its index increases. In the rest of this chapter, each content item $x$ is identified by its rank $r$, the values of the rank ranging from 1 to $X$.

**Modeling the geographical distribution of interests** The Zipf distribution does not account for the GDI. In the absence of real traces, a function that gives the number of caching locations in the network where each content item $r$ is requested is considered. Compared to previous methods that mainly focus on uniform distribution (e.g. [150][154]), the proposed model is designed to represent a wider range of conditions. More specifically, the function is so that the number of caching locations in the network where each content item is requested depends on the global popularity of each item. Intuitively, the more popular a content is, the more likely it is to be requested at a large number of different locations. Very unpopular contents, on the other hand, are more likely to be requested from more specific geographical locations. With the proposed function, the minimum number of locations where a content item can be requested is 1 and the maximum is the total number of caches in the network. In order to model various profiles of GDI (i.e. scenarios where most of the content items are requested from a large number of caching locations, or in contrast, from specific locations only), a parameter $\beta$ is introduced in the function. The parameter $\beta$ is a strictly positive constant that drives the characteristics of the GDI. The proposed function $f_\beta$ is then defined as follows:

$$f_\beta(r) = 1 + (M-1)\left(\frac{(X-r)}{(X-1)}\right)^\beta \tag{6.1}$$

where $M$ is the total number of caches in the network and $X$ is the total number of content items in the list. The profile of the function $f_\beta$ for three values of the parameter $\beta$ is depicted in figure (6.2) for $X = 110$ and $M = 11$.

As it can observed, when $0 < \beta < 1$, the function $f_\beta$ exhibits a concave profile, i.e. most of the content items are requested from a large number of caches. The lower the value of $\beta$ is, the

higher the number of caching locations for each content item is on average. The distribution of the interests is more homogeneous between the different caches, and the homogeneity increases as the value of $\beta$ decreases. When $\beta = 1$, the function $f_\beta$ is linear. The number of locations where each content item $r$ is requested linearly decreases with the global popularity rank of $r$. When $\beta > 1$, the function $f_\beta$ exhibits a convex profile, i.e. most content items are requested from a subset of locations only. The higher the value of $\beta$ is, the lower the number of caching locations for each content item is on average. In this case, the GDI is more heterogeneous and the heterogeneity increases as the value of $\beta$ increases. By tuning the value of the parameter $\beta$, different degrees of heterogeneity of the GDI can be modeled and, as such, the function $f_\beta$ can be used to represent a wide range of scenarios.

## 6.4 Placement Strategies

Two categories of content placement strategies that can be used by the cache managers in the access nodes are considered. The proposed strategies satisfy the following constraints:

1. There is at least one copy of each content item cached in the network.
2. A content item $r$ is copied in a cache only if there is enough caching capacity to accommodate $r$.
3. A content item $r$ is copied in a cache only if $r$ is not already cached at this location.

The strategies can be implemented in a decentralised fashion through the intelligent in-network substrate, which facilitates the exchange of information as explained in section 6.3.2. More specifically, the objective of the proposed strategies is to determine the number of copies of each content item to store in the network, as well as their placement at different possible caching locations. The placement decisions are taken by the cache managers that coordinate among themselves to decide upon the most appropriate caching configurations to apply. The cache managers do not independently decide how to use their associated caching space (i.e. the content items stored locally), but instead they communicate through the substrate to exchange local information, such as the content items which are already stored locally. The two categories of strategies differ both in terms of the volume of information required and in terms of the characteristics of the information to be exchanged between the managers.

The decision-making mechanism to support the proposed placement strategies can follow an iterative approach as proposed for instance in [80]. In this case, content placement decisions are taken iteratively, so that only one cache manager is permitted to take a placement decision at a time. The order followed by the managers is predetermined and is provided to each manager prior to the execution of the algorithm. For scalability purposes, a decision-making approach

that can parallelise the decisions taken by each manager can also be considered [151]. In this case, however, a set of carefully selected constraints will need to be implemented at each cache manager to avoid inconsistent configurations.

### 6.4.1 Local-Popularity driven Strategy

The local-popularity (LP) driven strategy follows a two-phase process. In the first phase, the cache managers collaborate to decide where to cache the first copy of each content item ensuring that the first constraint is satisfied. In the second phase, each manager independently profits from the potential remaining caching space in the associated cache to replicate some of the more locally requested content items not already cached, until the total local space is consumed. Replicating content items locally limits the number of redirected requests. Each cache manager maintains the list $\mathcal{L}_{\text{pop}}$ of its locally requested content items ordered by decreasing local popularity (i.e. local aggregated volume of requests for each content item). The list is provided to the managers at each reconfiguration cycle.

**First Phase:** Each content placement decisions taken by a manager consists in selecting a single content item, from the locally requested elements, to cache locally, so that the selected item a) has the highest local popularity, b) is not already cached somewhere else in the network, and c) there is enough space to store the content item. This procedure is executed until no further decision can be taken. In order to ensure that the first constraint is satisfied, a mechanism to check if all content items are cached is executed at the end of this phase. If not, non-cached content items are randomly placed in one of the caches with enough remaining capacity. This phase always terminates since, as explained in section 6.3.3, $V_{\text{X}} \leq V_{\text{M}}$ by design.

**Second Phase:** Each manager selects a set of content items to replicate among the locally requested elements (i.e. from $\mathcal{L}_{\text{pop}}$), so that the selected items are the locally most popular not already cached. The number of selected content items depends on the remaining caching space.

With this strategy, the placement decisions taken by each manager depend essentially on the local user demand. Apart from the first phase, where information about content items previously cached in the network is exchanged/used, the managers do not use any network-wide information to decide upon local placements. Each new local placement is advertised to all other managers to ensure a consistent view of which content items are currently cached in the network.

### 6.4.2 Global-Popularity driven Strategy

The global-popularity (GP) driven strategy also follows a two-phase process. While the first phase ensures that the first constraint is satisfied, the second phase allows cache managers to

replicate content items locally. Unlike the local popularity strategy, however, collaborative decisions between the cache managers extend to the second phase. Each manager maintains a copy of the list of content items requested in the network ordered by decreasing global popularity, $\mathcal{L}_{\text{pop}}$ and for each content item $r$, the list of caching locations where $r$ is requested ordered by decreasing number of requests, $\mathcal{L}^{\text{r}}_{\text{loc}}$. As such, all managers have the same global knowledge about content popularity and geographical distribution of interests in the network. The lists are provided to the managers at each reconfiguration cycle.

**First Phase:** The first phase of the GP strategy consists in iteratively considering each content item $r$ in the list $\mathcal{L}_{\text{pop}}$, in order to decide where to place the first copy of $r$. The cache with the highest aggregated number of requests for $r$ (and strictly greater than zero), and with sufficient space, is selected. If none of the caches can satisfy these conditions, the content item is temporarily disregarded and marked as *FAIL*. The procedure continues until all items in $\mathcal{L}_{\text{pop}}$ are considered. Once the procedure is completed, caches with sufficient capacity are randomly selected to store any content item marked as *FAIL*. In a similar fashion to the LP strategy, this phase always terminates since $V_{\text{X}} \leq V_{\text{M}}$ by design.

**Second Phase:** The procedure followed is similar to the one used in the first phase. The only difference lies in the number of caches selected to store a copy of each content item. Instead of selecting one cache, a pre-defined number of $n$ caches are selected, where $n$ is at most equal to the total number of caches in the network. If it is not possible to find $n$ caches satisfying the conditions, a copy of the content item is stored in the maximum number of possible locations. The process stops when it is not possible to find a new feasible placement. The number $n$ is an input of the algorithm and is provided to cache managers prior to the execution of the algorithm.

In this approach, placement decisions are driven by the global popularity of the content items. In addition, the algorithm tries to store each content item $r$ in the location where the demand for $r$ is the highest. The number of copies of each content item (i.e. replication degree) is driven by $n$. As $n$ increases, the globally more popular content items tend to be more replicated than the others, whereas the replication degree of each content item tends to be more uniform when $n$ is small. The manager decisions are coordinated throughout the process. The communication overhead incurred by the GP strategy is therefore larger than that of the LP strategy.

## 6.5 Evaluation

This section presents the evaluation of the performance of four strategies belonging to the two categories described in section 6.4 in terms of network resource utilisation according to various user demand characteristics. The four strategies are the following:

1. The local-popularity driven strategy, noted LPS.
2. The global popularity driven strategy with replication factor $n = 1$, noted GPS_1.
3. The global popularity driven strategy with replication factor $n = 2$, noted GPS_2.
4. The global popularity driven strategy with replication factor $n = 3$, noted GPS_3.

### 6.5.1 Experimentation Setup

The PoP-level Abilene network [99] was used to simulate the proposed caching infrastructure presented in figure (6.1). The network considered has 11 nodes (which are all source nodes) and 28 unidirectional links. In the simulation, one cache is associated with each of the network nodes and the same capacity is assigned to all links. The total caching space is set so that twice the total volume of content items to cache can be accommodated in the network, and is uniformly distributed between the 11 caching locations. It is worth noting that, unlike the other chapters of this thesis, this work considers the use of the Abilene network topology only. The objective of the evaluation was to investigate whether knowledge about user demand characteristics can provide useful information regarding the performance of a content placement strategy in terms of network resource utilisation. The influence of the network topology characteristics on the performance of a strategy would require an independent study and could be part of future research directions. The Java-software presented in previous chapters was extended to implement the proposed network infrastructure and placement strategies.

In the absence of real demand traces, synthetic user demand profiles were generated. As explained in section 6.3.3, each profile is characterised by the global content popularity distribution represented by the parameter $\alpha$, and by the geographic distribution of the interests represented by the parameter $\beta$, i.e. by a pair ($\alpha$, $\beta$). In order to evaluate a wide range of profiles, 20 values of parameter $\alpha$ with $\alpha \in [0.1; 2]$ ([162]), and 20 values of parameter $\beta$ with $\beta \in [0.2; 10]$, were considered, i.e. a total of 400 ($\alpha$, $\beta$) pairs. The values of $\alpha$ and $\beta$ are chosen to be representative of a wide range of possible conditions, and as such are likely to cover realistic scenarios as well. Each pair defines the total number of requests for each content item in the network, and for each content item $m$, the number of locations where $m$ is requested. 100 samples were generated for each pair, i.e. different locations from where each content item is requested were randomly selected, and as a result, a total of 40,000 evaluation samples was

obtained. It is worth noting that for comparison purposes, it was ensured that, for a given $\beta$ and a given sample, the locations from where each content item is requested was identical for all values of $\alpha$. The total volume of requests in the network was constant in all experiments. A list of approximately 100 content items to cache in the network was considered for the evaluation to be manageable in time. All experiments were performed on a laptop with a 2.80 GHz Intel Core i7-2640M processor and 8 GB memory.

### 6.5.2 Evaluation Methodology

In order to simulate various user demand profiles, different values of parameters $\alpha$ and $\beta$ were used. In practice, however, characterising the demand through $\alpha$ and $\beta$ may not be feasible, since it may be difficult to retrieve the actual value of the parameters given the monitored information in the network. Two metrics that can be used, instead, to characterise the user demand were therefore defined.

**Metric $H_{\mathbf{G}}$** The metric $H_{\mathrm{G}}$ describes the heterogeneity of the geographic distribution of interests for the different content items. Let $d_{\mathrm{mr}}$ represent the demand for content item $r$ at caching location $m$. Let $M$ be the total number of caches in the network, and $X$ the total number of content items. For each content item $r$, a parameter $\bar{h}_{\mathrm{g}}(r)$ that characterises the heterogeneity in terms of geographical distribution of interests (GDI) for content item $r$ in the network is defined as follows:

$$\bar{h}_{\mathrm{g}}(r) = \frac{\sqrt{\frac{1}{M}\sum_m \left(d_{\mathrm{mr}} - \mathbb{E}_{\mathrm{m}}\left(d_{\mathrm{mr}}\right)\right)^2}}{\sum_m d_{\mathrm{mr}}} \ . \tag{6.2}$$

where $\mathbb{E}_{\mathrm{m}}(d_{\mathrm{mr}})$ is the expectation of the demand for content item $r$ at each caching location $m$, i.e. $\mathbb{E}_{\mathrm{m}}(d_{\mathrm{mr}}) = \frac{1}{M}\sum_m d_{\mathrm{mr}}$. The parameter $\bar{h}_{\mathrm{g}}(r)$ represents the standard deviation of the demand for content item $r$ at each cache $m$ normalised by the total demand for $r$ in the network.

The metric $H_{\mathrm{G}}$ is then defined as the normalised sum of the $\bar{h}_{\mathrm{g}}(r)$:

$$H_{\mathrm{G}} = \frac{1}{X}\sum_r \bar{h}_{\mathrm{g}}(r) \ . \tag{6.3}$$

**Metric $H_{\mathbf{P}}$** To characterise the heterogeneity in terms of global popularity between the different content items in the network, the metric $H_{\mathrm{P}}$ is defined as follows:

$$H_{\mathrm{P}} = \frac{\sqrt{\frac{1}{X}\sum_r \left(\sum_m d_{\mathrm{mr}} - \mathbb{E}\left(\sum_m d_{\mathrm{mr}}\right)\right)^2}}{\sum_r \sum_m d_{\mathrm{mr}}} \ . \tag{6.4}$$

where $\mathbb{E}\left(\sum_m d_{\mathrm{mr}}\right)$ is the expectation of the total number of requests for a content item in the network, i.e. $\mathbb{E}\left(\sum_m d_{\mathrm{mr}}\right) = \frac{1}{X}\sum_r \sum_m d_{\mathrm{mr}}$ and $\sum_r \sum_m d_{\mathrm{mr}}$ is the total volume of

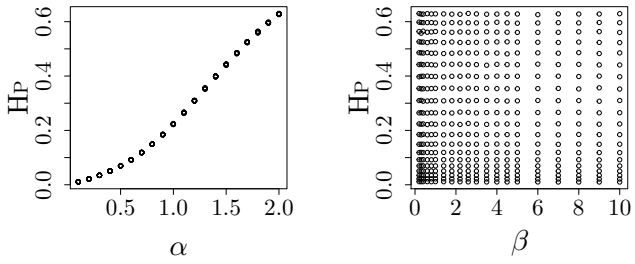



Figure 6.3: Value of $H_{\mathrm{P}}$ vs. parameter $\alpha$ and parameter $\beta$.

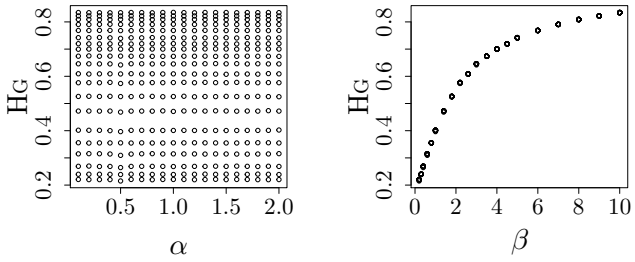



Figure 6.4: Value of $H_{\mathrm{G}}$ vs. parameter $\alpha$ and parameter $\beta$.

requests in the network. Therefore, parameter $H_{\mathrm{P}}$ represents the standard deviation of the total number of requests for each content item in the network normalised by the total volume of requests in the network.

As can be noticed, the value of $\alpha$ and $\beta$ is not required to determine the value of $H_{\mathrm{G}}$ and $H_{\mathrm{P}}$. These are obtained through the knowledge of the total number of content items to cache, the total number of caches in the network and the demand for each content item at each caching location only. These can be available in the network, either in the form of static parameters (e.g. number of caches), or through monitoring information (e.g. aggregated user demand at the network edges). In order to observe the relationship between the two metrics and the parameters $\alpha$ and $\beta$, the values of $H_{\mathrm{P}}$ and $H_{\mathrm{G}}$ obtained for each of the evaluation samples against the different values of $\alpha$ and $\beta$ are plotted in figures (6.3) and (6.4).

As it can be observed in the figures, $H_{\mathrm{P}}$ is correlated to parameter $\alpha$ but is independent of parameter $\beta$. On the contrary, $H_{\mathrm{G}}$ is independent of parameter $\alpha$ but is correlated to parameter $\beta$. More specifically, the value of $H_{\mathrm{P}}$ strictly increases with the value of $\alpha$, and, as such, a larger value of $H_{\mathrm{P}}$ represents a higher degree of heterogeneity between the global popularity of content items in the network. In a similar fashion, the value of $H_{\mathrm{G}}$ strictly increases with the value of $\beta$, and, as such, a larger value of $H_{\mathrm{G}}$ represents a higher degree of geographic heterogeneity (i.e. GDI) in the network. It is worth mentioning that although there exists a correlation between the values of $H_{\mathrm{P}}$ and $H_{\mathrm{G}}$ and the parameters $\alpha$ and $\beta$, the formulae used to determine the actual values of the two metrics are independent of $\alpha$ and $\beta$. As such, these metrics are not adapted to the current settings only, which makes them valuable to analyse the popularity characteristics

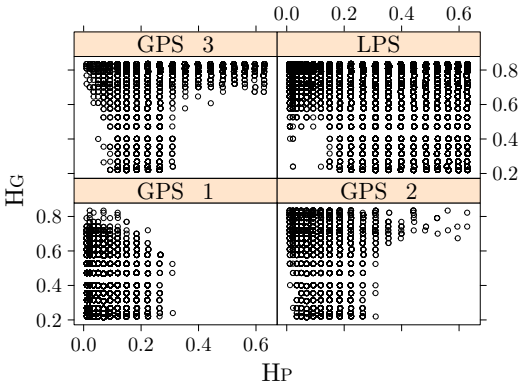



Figure 6.5: Performance of the four strategies according to different values of $H_{\mathrm{P}}$ and $H_{\mathrm{G}}$.

in any content distribution scenario. In addition, these can be directly applied to the monitored data since only the demand for each content item is required to compute the values of these metrics.

### 6.5.3 Results and Analysis

The performance of each of the four strategies in terms of the maximum link utilisation (max-u) that resulted in the network for different demand profiles are compared. A demand profile is characterised by a pair of values ($H_{\mathrm{P}}$;$H_{\mathrm{G}}$). The best strategy is the one that results in the lowest max-u.

Figure (6.5) shows the results for each of the 40,000 samples. For readability purposes, the figure is divided in four windows, one for each strategy. For each sample the best strategy is determined by comparing the different max-u. Each point in each strategy window means that this is the best strategy for the relevant evaluation sample. It is important to note that according to the experimental settings, each pair is associated to 100 samples, and as such, one point in a graph may actually represent several samples. As it can be observed from the figure, the performance of each strategy follows different trends. In the region defined by the smallest values of $H_{\mathrm{P}}$ and $H_{\mathrm{G}}$, strategy $GPS\_1$ is exclusively the best strategy. On the contrary, in the region defined by a value of $H_{\mathrm{P}}$ greater than $0.25$ and value of $H_{\mathrm{G}}$ smaller than $0.7$, the best strategy is always strategy $LPS$. For other values of $H_{\mathrm{P}}$ and $H_{\mathrm{G}}$, however, determining which strategy can achieve the best performance depends on the sample considered, and as such, no strategy can be exclusively identified as the best strategy.

Due to the replication logic, all content items tend to have a similar replication degree with strategy $GPS\_1$. As it can be observed from the results, such a strategy is preferable when both the GCP and the GDI are more homogeneous, i.e. when the values of $H_{\mathrm{P}}$ and $H_{\mathrm{G}}$ are low. In this case, using a strategy that partitions the overall caching space in a fair

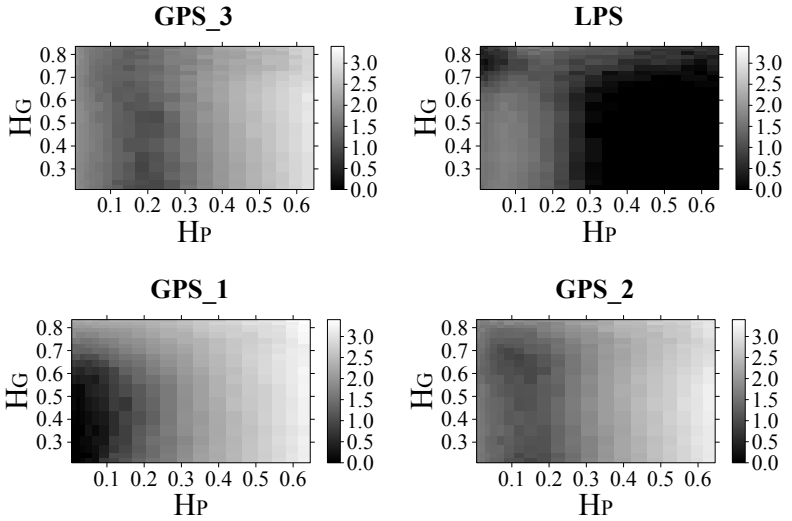



Figure 6.6: Density graphs of the average deviation of the maximum utilisation for each strategy.

manner between the content items can lead to better network resource utilisation. Compared to strategy $GPS\_1$, the content replication degree is less uniform between the content items with strategy $GPS\_2$ and $GPS\_3$. Given that the speed at which the overall caching space is consumed is driven by the factor $n$, as defined in section 6.4.2, globally popular content items have a significantly higher replication degree than other content items, the difference increasing as factor $n$ increases. When the value of $H_{\mathrm{P}}$ is low but the value of $H_{\mathrm{G}}$ increases (i.e. the GDI becomes more heterogeneous), a placement strategy that behaves in a more discriminating fashion in terms of replication degree achieves better performance. Due to the absence of coordination during the second phase of strategy $LPS$, globally popular content items tend to be significantly more replicated than the others. As it can be observed, for large values of $H_{\mathrm{P}}$, a strategy that aggressively increases the replication degree of the most popular content items leads to better network resource utilisation. In this case, the volume of traffic incurred in the network is mainly dominated by the requests for these content items. By replicating these content items in most locations, a large volume of traffic can thus be removed from the network, which leads to lower max-u.

In order to see how the performance of the different strategies compares to each other, the average deviation of the max-u obtained by each strategy with the minimum max-u reported for each ($H_{\mathrm{P}}$,$H_{\mathrm{G}}$) pair is also analysed. The average deviation is determined by computing, for each strategy, the deviation from the minimum for each sample of a given ($H_{\mathrm{P}}$,$H_{\mathrm{G}}$) and by averaging the results over the total number of samples considered. For each strategy, the values obtained are depicted in a density graph, as shown in figure (6.6). The magnitude of the deviation is represented by different shades of grey, the darkest shade corresponding to a deviation of 0%

and the lightest to the maximum deviation observed in the experiments. Given that the values obtained spanned a large range, a logarithmic scale was used to measure the deviation. As can be observed, strategy $LPS$ performs uniformly well for $H_{\mathrm{P}}$ greater than $0.3$ and $H_{\mathrm{G}}$ smaller than $0.7$ since the average deviation is close to $0$. In contrast, the three other strategies obtain poor performance on average in that region. Very good performance is also obtained by strategy $LPS$ for low values of $H_{\mathrm{P}}$ and large values of $H_{\mathrm{G}}$. It can also be noticed that strategy $GPS\_1$ performs uniformly well for the lowest values of $H_{\mathrm{P}}$ and $H_{\mathrm{G}} < 0.5$. Whereas the results in figure (6.5) indicate that the lowest max-u can be achieved with strategies $GPS\_2$ and $GPS\_3$ in some cases, it can be observed in figure (6.6) that these strategies lead on average to much higher resource utilisation. The performance is therefore less predictable.

The performance of each strategy in terms of resource utilisation is influenced by the user demand. More precisely, the evaluation results show that, in some cases, knowing $H_{\mathrm{P}}$ and $H_{\mathrm{G}}$ is not enough to determine which strategy to apply since the best performance can be achieved by several of them. It is possible, however, for certain values of $H_{\mathrm{P}}$ and $H_{\mathrm{G}}$, to identify one strategy that outperforms the others. As such, the proposed metrics can provide useful indications regarding the performance one strategy can achieve over another and, as such, can be used by the ISP to improve the utilisation of network resources.

## 6.6 Conclusion

This chapter describes a cache management scheme by which simple and lightweight content placement strategies can be used by an ISP to determine the placement of content items in the various network caching points according to the characteristics of user demand. Although the proposed ISP caching solution may not provide the level of service that CDNs do, it is low-cost compared to traditional CDN services which (i) require a lot of power for operating and cooling the storage infrastructure, (ii) have collocation costs, and (iii) require human supervision. In contrast, the proposed approach is automated, has smaller distributed storage, and has simplified functionality that can be realised by commodity hardware boxes. The deployment of such an approach can allow ISPs to manage their network resources more effectively. This can subsequently improve user experience which could work towards the benefit of some CDN providers.

This work investigated and proposed a methodology to evaluate the influence of user demand on the performance of different categories of strategies to manage the placement of content items in the various network caching locations. The proposed strategies differ in terms of the volume and nature of the information required to determine the new caching configurations.

In order to evaluate the influence of user demand profiles on the performance of the proposed strategies in terms of network resource utilisation, different metrics were developed to model the content request distribution. In particular, a new model was designed to represent the geographic distribution of user interests in the network. The proposed metrics are designed so that they can be directly applied to monitored data, which make them valuable to analyse the popularity characteristics in any content distribution scenario. In addition, the results demonstrate that the proposed metrics can provide useful indications regarding the performance one strategy can achieve over another and, as such, can be used by an ISP to improve the utilisation of network resources.

## Chapter 7

# Control Loop Interactions Management

## 7.1 Introduction

In previous chapters, the proposed resource management framework was applied to three different resource management applications and a specific control loop functionality was developed for each of them. However, a self-management solution with rich functionality would require multiple control loops operating at the same time, which can interact with each other. This raises important research issues as these interactions can cause configuration instabilities and subsequently, network performance degradation, especially in the presence of contradicting control loop objectives. Developing mechanisms to manage control loop interactions is therefore essential.

This chapter considers the problem of interactions in a scenario where two control loops that perform dynamic network resource reconfiguration for load balancing and energy management purposes, respectively, are implemented in the network. The objective of the load balancing application (chapter 4) is to distribute the traffic load in the different parts of the network so that the maximum link utilisation can be minimised. In contrast, the energy management application (chapter 5) aims at minimising the number of links that are needed to support the traffic load in order to reduce the energy consumption in the network. Given the orthogonal nature of the two optimisation objectives, inconsistent configuration decisions are likely to be enforced when the two control loops operate in parallel.

The work presented in this chapter investigates how the resource management framework developed in this thesis can be used to coordinate the decision-making processes executed by the two applications so that acceptable network performance can be achieved in terms of resource utilisation and energy consumption. The proposed solution aims at decoupling control loop operations by treating each of them sequentially. The actual management mechanism is executed by the network ingress nodes through the management substrate. The approach is

evaluated using real network topologies and traffic traces, and the performance results are compared against those obtained with each control loop operating independently. These show that a trade-off can be achieved between the two optimisation objectives.

The rest of this chapter is organised as follows. Section 7.2 presents related work. Section 7.3 analyses how potential configuration inconsistencies can arise and describes the proposed interaction management solution. The principles of the resource management scheme developed to satisfy the two optimisation objectives are detailed in section 7.4. The performance evaluation of the approach is presented in section 7.5. Finally, a summary of the work is presented in section 7.6.

## 7.2 Related Work

The importance of coordination models to manage the coexistence of multiple autonomic elements has been highlighted since the early work on autonomic computing [4][27], which defined these issues as key research challenges for the development of self-managed systems.

Most of the previous work in this area has focused on developing generic architectures to manage the interactions between multiple control loops, e.g. [163][164][165]. In fact, it was argued in [166] that understanding the structural arrangements according to which control loops operate was essential in order to address the engineering issues associated with the coexistence of multiple loops. The work identified different models that could be considered when implementing self-management modules, such as sequential, parallel and hierarchical. According to the authors, several parameters can influence the type of models to use, such as, for instance, the timescale of operation or the type of actions to perform.

A hierarchical architectural model was proposed in [163] to manage control loop interactions. In this work, each self-management application is represented as an autonomic element component. The management decisions of the different autonomic elements are controlled by an arbiter element, which is elected among the autonomic elements to act as a coordinator. The arbiter is responsible for determining which actions to perform based on current system/network conditions. A policy-based hierarchical framework was also proposed in [164]. In this work, the control actions are identified based on their timescale and locality, and the coordination between the control loops is realised through a centralised application controller element. A centralised solution was also considered in [165], where the authors envisioned the role of the centralised controller component as an enabler for consistent information sharing, decision-making and reasoning processes executed by independent self-managed applications. All these approaches assume that precise models of the managed system and of the effects of management decisions

can be used in order to determine the best sequence of actions to apply. In practice, however, this may not always be possible, especially in the case of complex network environments with rich functionality.

A key issue when managing multiple independent control loops concerns the harmonisation of the decisions taken by each control loop so that individual objectives can be satisfied. More specifically, the resulting configuration should guarantee the overall system stability, while ensuring that acceptable levels of performance are maintained for each control loop. From a mathematical point of view, satisfying simultaneously multiple objectives is the subject of multi-objective optimisation problems. These aim at determining optimal decisions in the presence of trade-offs between two or more conflicting objectives. The use of multi-objective optimisation techniques to solve engineering problems has received a lot of attention from the research community [167]. A multi-objective optimisation-based approach was recently presented in [168] in the context of object search optimisation in mobile social networks. In this work, the authors tried to optimise three conflicting objectives (i.e. minimising energy consumption, minimising search delay and maximising query recall) and used a genetic algorithm to solve the problem. An approach based on game-theory was considered in [78] to harmonise the decisions of a traffic engineering application (i.e. load balancer) and a content request rerouting application (i.e. server selection). The use of game-theory principles was also investigated from a theoretical perspective in [169] in the context of the coexistence of load balancing and energy management applications. This analysed the mathematical conditions under which a fair trade-off can be achieved between the two control loop objectives. In contrast, the work presented in this chapter focuses on more practical issues.

The problem of interactions between traffic engineering and energy management functions was also considered by Athanasiou et al. in [133]. In that work, the authors assume that the level of energy consumption in the network at different times of the day is predetermined by the network operator. In order to satisfy the energy objective, they propose a load balancing-aware energy management application that uses the network configuration obtained as a result of a load balancing algorithm to switch off network links. This differs from the approach described in this chapter that considers both applications independently.

## 7.3 Control Loop Interactions

### 7.3.1 Adaptive Network Resource Management Scenario

In contrast to previous chapters, in which the management substrate was used in the context of independent resource management applications, a more sophisticated scenario is considered

here. It is assumed that the two resource management functionalities described in chapters 4 and 5, namely the load balancing control loop (LB-CL) and the energy management control loop (EM-CL), are implemented in the network. Based on path diversity provided by multi-topology routing [117], the reconfiguration decisions of both loops concern the value of traffic splitting ratios applied at network ingresses, so that individual objectives can be met.

Exploiting the fact that many links in core networks are bundles of multiple physical cables [138], the objective of the energy management approach is to offload as many router line cards as possible, which can subsequently enter sleep mode. Line cards can be full if their load is equal to their capacity, utilised if their load is not zero and less than their capacity, and non-utilised if they have zero load. As described in chapter 5, one of the key decisions at each iteration of the reconfiguration process concerns the bundled link to consider for (a) removing traffic from, and (b) assigning that traffic to. This decision is based on a ranked list of all utilised line cards in the network according to their load. Traffic load is iteratively moved from the least loaded utilised line card to more loaded ones that can accommodate this load and thus potentially fill up their remaining capacity.

In the case of load balancing, the objective is to permanently balance the load in the network by moving some traffic away from highly utilised links towards less utilised ones in order to reduce the utilisation of hotspots against dynamic traffic behaviours. In order to minimise the maximum utilisation in the network, the algorithm iteratively adjusts the splitting ratios of the traffic flows, so that traffic can be moved away from the link with the maximum utilisation, $l_{\max}$. The updated splitting ratios need to ensure that the utilisation of $l_{\max}$ is reduced while no other link in the network obtains a new utilisation higher than the original utilisation of $l_{\max}$.

To prevent inconsistencies due to concurrent traffic splitting adjustments among multiple substrate nodes executing a control loop, at each reconfiguration iteration, the nodes coordinate through the substrate to select one of them (called the Deciding Entity) that will compute new splitting ratios over its locally originating traffic flows. This logic is represented by a rule that allows the selection of the node which is most likely able to determine a configuration that satisfies the objective of the control loop.

It should be noted that, for consistency between the two functionalities, the network model used in this work is the one considered in chapter 5, i.e. network links are bundles of multiple physical cables.

### 7.3.2 Problem Statement

The objective of the LB-CL and the EM-CL is to adjust the traffic splitting ratios of incoming flows (i.e. control parameters) in order to satisfy their own optimisation objectives. In the case

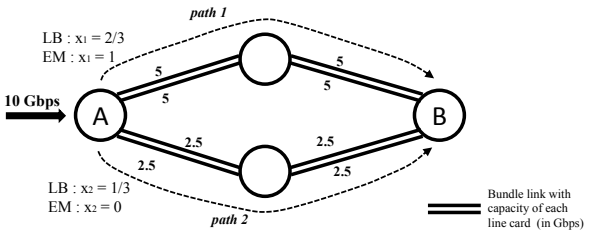



Figure 7.1: Example of inconsistent network configurations.

of the LB-CL, the objective is to spread the traffic load in the network, so that the maximum link utilisation (max-u) can be minimised. In contrast, the EM-CL tries to concentrate the traffic load onto a subset of links so that non-active line cards can be turned off. As such, the decisions taken by each control loop individually may result in inconsistent network configurations. An example of such an inconsistency is illustrated on a simple scenario in figure (7.1). In this example, the LB-CL and the EM-CL need to decide how to split the incoming traffic flow at node A over the two paths $p_1$ and $p_2$ to the destination node B to optimise their own objective. In order to minimise the max-u, the LB-CL decides to split the traffic proportionally to the bottleneck link capacity in each path, which gives the ratios $\frac{2}{3}$ and $\frac{1}{3}$, for path $p_1$ and $p_2$, respectively. This gives a max-u equal to $66\%$ and a total number of active line cards equal to $8$. In order to minimise the total number of active line cards, the EM-CL decides to concentrate the traffic load on path $p_1$, i.e. all traffic is routed over $p_1$. This configuration leads to a max-u of $100\%$ and a total number of active line cards equal to $4$. As such, the decisions of one control loop can adversely affect the performance of the other. In the example, the network configuration computed by the EM-CL results in an increase in max-u of almost $40\%$, whereas the optimal configuration of the LB-CL leads to an increase of $50\%$ in terms of the number of active line cards.

More generally, inconsistent configurations are the result of the interactions between the control loops. The LB-CL and the EM-CL present an overlap in terms of the resources affected by their reconfiguration decisions (i.e. link utilisation), and as such, their decisions are not independent. Interactions can also be identified according to the timescale at which control loops operate. Coexisting control loops can, for instance, operate at different timescales. In this case, hierarchical structures by which one control loop provides guidance for the operation of the other, are usually considered, e.g. [166]. In this scenario, however, the LB-CL and the EM-CL operate at the same frequency and, as such, do not follow any hierarchy. Due to the overlap in time, the objective of one control loop can severely suffer from the decisions taken by the other. In realistic network scenarios, where the splitting ratio adjustments of multiple

traffic flows need to be considered during the same reconfiguration interval, the existence of two interacting control loops may lead to a significant number of conflicting decisions. These can cause network instabilities and, as a result, adversely affect the overall performance. The management of control loop interactions is therefore essential.

### 7.3.3 Proposed Solution

To manage the interactions between the LB-CL and the EM-CL, an approach that sequentially optimises the two objectives is proposed. This approach aims at harmonising the reconfiguration decisions by preventing the two control loops from operating simultaneously. More specifically, the Deciding Entity node selected at each reconfiguration iteration is responsible for executing successively each control loop according to a predetermined order, so that the splitting ratios of the local traffic flows can be readjusted to satisfy both objectives. However, instead of optimising each objective independently, the constraints of both problems are considered when executing each control loop. This facilitates improvement of the objective of one control loop while maintaining some level of performance for the other, thus avoiding performance degradation.

A natural way to implement the proposed approach is to replace the LB-CL and EM-CL by a new control loop, which combines the objectives and constraints of the two applications. This solution has, however, some limitations especially in terms of flexibility. By design, such a global control loop highly depends on the applications considered. In order to introduce an additional control loop in the system or to modify an existing one, the network operator would have to modify the global loop, which makes the implementation of any changes harder.

To overcome these limitations, a modular structure that relies on the architectural model presented in figure (7.2) is proposed to implement the two-objective resource management approach. As depicted in the figure, a set of elements needs to be deployed at each network edge node[1]. The LB-CL and the EM-CL are implemented as two separate entities (or algorithms), each of them containing the necessary logic to achieve the relevant objective. The interaction between the two loops is controlled through the *control loop coordinator* entity, which implements the logic to perform the proposed sequential optimisation process. The control loop entities do not directly interact with each other, but instead communicate with the coordinator. The latter provides to each loop network statistics information (e.g. link utilisation), as well as a set of constraint parameters to be enforced by each control loop in order to satisfy the global resource optimisation objective. The control loops, in return, send the outcome of the reconfiguration (i.e. the newly computed splitting ratios) back to the coordinator.

[1]The model is an extension of the architecture presented in chapter 4 and illustrated in figure (4.3).

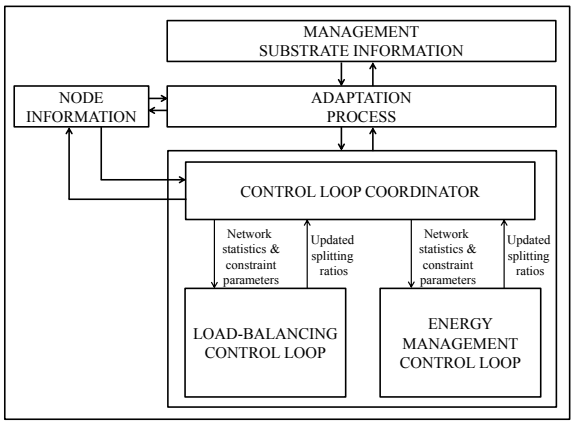



Figure 7.2: Architecture of a network edge node.

Compared to the global design approach, the proposed implementation offers more flexibility to apply changes. With this structure, the addition of a new control loop in the system requires the introduction of a new algorithm component at each network edge node and the adaptation of the control loop coordinator, but does not affect any other existing application.

## 7.4 Resource Management Approach

### 7.4.1 Adaptation Process

In a similar fashion to the approach used for load balancing and energy-saving in chapters 4 and 5, respectively, the resource management mechanism relies on an adaptation process managed by the intelligent in-network substrate and performed periodically in short timescales. It consists of a sequence of reconfiguration actions decided in a coordinated fashion between the nodes in the substrate that adjust the splitting ratios of network traffic flows. To prevent inconsistencies between concurrent traffic splitting adjustments, only one substrate node is permitted to take reconfiguration decisions at a time. At each iteration of the adaptation process, one node in the substrate is selected to be the Deciding Entity, i.e. to perform some reconfigurations over its locally originated traffic flows. The mechanisms to select a unique Deciding Entity are implemented in the *Adaptation Process* component (figure (4.3)). These depend on the structure of the substrate and are similar to those described in chapter 4.

### 7.4.2 Reconfiguration Process

The Deciding Entity node selected at each iteration of the adaptation process executes locally a reconfiguration process, which aims at adjusting the splitting ratios of local traffic flows in order to satisfy the two resource optimisation objectives. The reconfiguration process is a sequential process in which the control loops are executed iteratively by the selected Deciding Entity

according to a predetermined order common to all substrate nodes. The process is orchestrated by the coordinator entity, as illustrated in figure (7.2).

In order to maintain acceptable levels of performance with respect to each optimisation objective, each loop needs to satisfy the constraints of each other. In the case of the LB-CL, a new configuration (i.e. new splitting ratios) is accepted if the utilisation of the link $l_{\text{max}}$ with the maximum utilisation can be reduced while no other link in the network obtains a new utilisation higher than the original utilisation of $l_{\text{max}}$. The utilisation of each network link $l$ after the reconfiguration satisfies the following constraint:

$$\mathbf{C}_{\text{LB}}\textbf{:} \qquad \forall l, \quad u(l) \leq u(l_{\text{max}}). \tag{7.1}$$

In the case of the EM-CL, the new configuration is accepted if the action of diverting traffic away from the selected link $l_{\text{LCU}}$ with a utilised line card, does not activate a new line card in the network. This implies that the load $\rho(l)$ of each network link $l$ needs to be lower than an upper bound value $\rho_{\text{lim}}(l)$ defined as the product of the current number of active line cards in the link by the capacity of the line cards[2], i.e.

$$\mathbf{C}_{\text{EM}}\textbf{:} \qquad \forall l, \quad \rho(l) \leq \rho_{\text{lim}}(l). \tag{7.2}$$

As such, the constraints $\mathbf{C}_{\text{LB}}$ and $\mathbf{C}_{\text{EM}}$ influence the volume of traffic that can be diverted towards any network link. Based on current network statistics and splitting ratios configuration, the coordinator entity computes the maximum volume of traffic that can be diverted towards each network link so that both constraints can be satisfied. This is encoded in a *Link Constraint Table* structure (<link, volume>), which is provided to each control loop component as an input. More specifically, the coordinator entity initially provides the table to the first control loop to execute. Upon receiving the information, the control loop triggers its reconfiguration algorithm with the objective to adjust the splitting ratios of local traffic flows so that all constraints encoded in the *Link Constraint Table* can be satisfied. The outcome of the algorithm (i.e. new splitting ratios configuration) is then returned to the coordinator entity, which updates the values in the *Link Constraint Table*. The new table is passed to the second control loop which executes its reconfiguration algorithm. At the end of the process, the new configuration is enforced by the coordinator. The main actions executed by the coordinator entity are summarised in figure (7.3).

### 7.4.3 Time Complexity

Given the sequential nature of the approach, the convergence time of the proposed reconfiguration process is affected by (i), the number of control loops implemented in the system, and (ii),

[2]In this work, it is assumed that all line cards in a bundle have the same capacity.

**Main actions executed by the coordinator entity**

i. Compute the Link Constraint Table.
ii. Send the Link Constraint Table to the first control loop entity.
iii. Get the updated traffic splitting ratios from the first control loop entity.
iv. Update the entries in the Link Constraint Table.
v. Send the Link Constraint Table to the second control loop entity.
vi. Get the updated traffic splitting ratios from the second control loop entity.

Figure 7.3: Main actions executed by the coordinator entity.

the time complexity of each resource management scheme. It was shown in chapters 4 and 5 that the execution time of the reconfiguration algorithm used in both applications is defined by the number of local traffic flows. In the case a PoP-level network topology with $N$ nodes, it is in the order of $O(N^2)$.

## 7.5 Evaluation

### 7.5.1 Experimentation Setup

The performance of the proposed approach was evaluated based on the two PoP-level networks Abilene [99] and GEANT [100], for which traffic traces covering a period of 7 days were considered. The configuration of the bundled links in the two physical topologies is similar the one presented in Table (5.3) in chapter 5. The 28 unidirectional BLs of Abilene are implemented with 112 LCs and the 68 BLs of GEANT with 210 LCs. The multi-topology routing planes (four in total) were computed according to the algorithm described in chapter 4 section 4.3.3. This section presents the results from the experiments in terms of energy consumption (i.e. percentage of active line cards in the network) and maximum utilisation observed over time in each network.

### 7.5.2 Performance Evaluation

In order to evaluate the performance, four cases were considered as follows.

- the energy management scheme (EM) described in chapter 5 with changing splitting ratios that achieve the energy-saving objective.
- the load balancing management scheme (LB) described in chapter 4 with changing splitting ratios that achieve the load balancing objective.
- the proposed two-control loop management scheme with the LB algorithm executed first, followed by the EM algorithm (LB-EM).

| | GEANT | | | Abilene | | |
|---|---|---|---|---|---|---|
| | **LB** | **EM-LB** | **LB-EM** | **LB** | **EM-LB** | **LB-EM** |
| **Average** | 36.04 | 19.67 | 19.72 | 20.25 | 11.38 | 14.20 |
| **Minimum** | 8.43 | 1.36 | 1.36 | 0 | 0 | 0 |
| **Maximum** | 54.16 | 32.07 | 29.62 | 40.03 | 35.12 | 35.12 |

Table 7.1: Percentage deviation in terms of active line cards.

| | GEANT | | | Abilene | | |
|---|---|---|---|---|---|---|
| | **EM** | **EM-LB** | **LB-EM** | **EM** | **EM-LB** | **LB-EM** |
| **Average** | 50.67 | 30.82 | 25.11 | 35.48 | 18.76 | 15.05 |
| **Minimum** | 12.60 | 5.01 | -1.23 | 2.46 | -0.81 | 0.73 |
| **Maximum** | 75.12 | 53.97 | 47.22 | 50.98 | 24.96 | 32.56 |

Table 7.2: Percentage deviation in terms of maximum utilisation.

- the proposed two-control loop management scheme with the EM algorithm executed first, followed by the LB algorithm (EM-LB).

In all cases, adaptation is performed every 15 minutes in both the Abilene and GEANT topologies with a maximum of 50 reconfiguration algorithm iterations. For each of the schemes, the percentage of line cards used to accommodate the traffic is measured, as well as the maximum utilisation in the network (max-u).

The performance of the EM scheme when executing in isolation (chapter 5) is used as a reference to compare the performance of each approach in terms of energy consumption (i.e. number of active line cards). The percentage of deviation from the EM scheme obtained with the LB, the EM-LB and the LB-EM schemes is reported in Table (7.1). The highest average deviation is obtained with the LB scheme in both networks. In addition, it can be observed that the LB-EM scheme and the EM-LB scheme achieve similar performance in the case of the GEANT network. However, the EM-LB scheme performs on average slightly better than the LB-EM in the case of Abilene (an average deviation of 11.38% and 14.20%, respectively).

A similar analysis is performed with respect to the maximum link utilisation. In that case, the LB scheme is used as the reference to compare the performance of each approach in terms of max-u. The percentage of deviation from the LB scheme obtained with the EM, the EM-LB and the LB-EM schemes is presented in Table (7.2). As expected, the highest average deviation is obtained with the EM scheme in both networks. Furthermore, it can be seen that there is no substantial difference between the performance of the EM-LB scheme and the LB-EM scheme.

Several observations can be made from these results. First, it can be noted that the order according to which the proposed two-objective management approach considered the two

control loops does not affect the performance. In addition, the results indicate that while each single loop application (i.e. the EM and LB schemes) performs the best with respect to its own optimisation objective, it always obtains the highest deviation with respect to the other objective. In contrast, the EM-LB scheme and the LB-EM scheme can always achieve intermediate performance, which shows that they can offer a trade-off between the two optimisation objectives.

### 7.5.3 Discussion

The evaluation shows that, despite the existence of contradicting objectives, a trade-off in terms of performance obtained for each application can be achieved with the proposed approach. The computation of well-engineered routing planes, coupled to the application of strong constraints to each algorithm, are essential requirements to obtain such results. However, given that the control loops are considered iteratively, no guarantee can be provided regarding the fairness of the solution. Furthermore, the performance may also be affected by the number of control loops in the system. An alternative solution could consider an approach based on joint optimisation. In that case, the reconfiguration decisions are harmonised by jointly optimising the different control loop objectives so that they can all benefit from the adaptation. Several techniques for the joint optimisation of multiple objectives have been proposed in the literature [167]. The common principle of these techniques is to determine a trade-off between each of the objectives and compute the configuration (i.e. the value of the parameters of the problem) that can satisfy this trade-off. A trade-off can be applied, for instance, by considering a weighted sum of each of the objectives. In this case, the weight of each objective is static and represents the relative "importance" of this objective compared to the others. A trade-off can also be defined through the introduction of priorities between the objectives. The objective with the highest priority is optimised while the others are considered as constraints. In that case, the resulting configuration allows the achievement of optimal performance for the highest priority objective, while guaranteeing only acceptable level of performance for the others.

## 7.6 Conclusion

This chapter investigates the important issue of control loop interactions in the context of the two control loops for adaptive load balancing and energy management. This is a complex problem for which an initial, and yet encouraging, solution is proposed. More specifically, the proposed approach allows each resource management application to achieve a performance trade-off between its own optimisation objective and the objective of its competitor. This can prevent performance degradation of individual applications that can occur in the presence of

conflicting resource management objectives. The approach is implemented using the proposed resource management framework and relies on mechanisms that are similar to the ones developed and validated for the load balancing and energy management applications.

## Chapter 8

# Conclusions and Future Research Directions

## 8.1 Summary

Current practices for the configuration of Internet Service Provider networks mainly rely on offline predictive approaches, with management systems being external to the network. These are incapable of maintaining optimal configurations in the face of changing or unforeseen traffic demand and network conditions, and, due to their rigidity, they cannot easily support the requirements of emerging applications and future network operations. Self-management has been proposed as a potential solution to these challenges bringing intelligence into the network and thus enabling customised management tasks in a flexible and adaptive manner.

The work presented in this thesis investigated how self-management principles can be extended and applied to fixed networks such that continuous resource optimisation can be supported through adaptive resource management functionality. Based on representative application scenarios, specific approaches have been developed to meet the requirements of the future Internet, focusing on the self-optimisation of network resources.

In order to support self-management functionality, a novel resource management framework was designed. Resource management approaches usually rely on centralised solutions executed by a manager that has a global view of the current conditions to periodically compute new configurations according to dynamic traffic behaviour. Although relatively simple to implement, centralised solutions have limitations in practice, especially in terms of scalability (i.e. communication overhead between the central manager and devices at runtime) and lag in the central manager reactions that may result in sub-optimal performance. In order to overcome these limitations, the proposed framework relies on a decentralised infrastructure, where management decisions are made by the network ingress nodes themselves. The necessary logic to realise the self-management functionality is encapsulated in a management substrate, which is a logical structure formed by the ingress nodes of the network. Nodes collaborate through the

substrate to decide on the sequence of actions to apply in order to better utilise the network resources.

This work investigated different models to organise the nodes in the substrate and developed several algorithms to compute the associated topology structures. The use of the proposed management infrastructure was exemplified based on three realistic resource management application scenarios, namely adaptive traffic engineering, energy efficiency and cache management, for which relevant solutions were developed. In addition, it tackled the important research issue of the management of control loop interactions in the context of the two control loops for traffic load balancing and energy management.

The performance evaluation showed that the proposed resource management schemes can help Internet Service Providers to manage their network resources in a more flexible and controllable manner. The decentralised nature of the proposed management framework avoids the scalability problems encountered by centralised approaches. In addition, the efficiency of the heuristics employed by the different management schemes make them suitable for adaptive reconfiguration processes. Finally, the logic to realise the resource optimisation objectives needs to be implemented in the ingress nodes of the network only, which simplifies the deployment of the proposed framework to a certain extend.

## 8.2 Future Research Directions

Potential future research directions of the work presented in this thesis can be summarised as follows.

**Energy management:** Although the evaluation results demonstrate that a substantial energy gain can be achieved with the proposed energy-saving resource management approach, future work could investigate how performance could be further enhanced. This could consider, in particular, complementing the proposed approach with an offline one (e.g. [77]) that computes the different routing planes for different intervals of the day (e.g. reduced topologies for off-peak times). Furthermore, it would be interesting to see whether additional energy gain could be obtained by controlling individual ports on router line cards. The energy savings are currently measured according to the number of active line cards in the network. In future work, precise models that can relate the traffic that passes through a router to the power consumption of the device could be developed. This would allow one to quantify the cost of data packets, and as such, could provide the Internet Service Provider with information about the cost associated with the data carried in its network.

**Content placement management:** In order to better understand the influence of user demand characteristics, this thesis assumes uniform content and cache size. In future work, it would be interesting to analyse the influence of different content and cache sizes on the performance of the proposed content placement strategies. In addition, future work could apply the proposed scenario to different network topologies. This work assumes that content popularity can be determined by the Internet Service Provider (ISP) and subsequently provided as input to the proposed placement algorithms. In practice, however, determining this information without input from the Content Delivery Network (CDN) is a key issue which could be further investigated. More generally, the cooperation model between the ISP and CDN providers is an issue that relates to the evolution of CDNs and the increased participation of ISPs in the delivery chain. In future extensions of this work, additional business models for the interaction between the two could be investigated.

**Control loop interaction management:** In order to avoid potential conflicts that may arise from the decisions made by the load balancing and the energy management applications, the approach proposed in this Ph.D. work is that control loops are executed in a sequential manner. Although the results demonstrate that a certain trade-off in terms of performance can be achieved for the objective of each control loop, fairness cannot be guaranteed since a notion of priority between the different applications is implicitly introduced. In addition, the complexity of the proposed approach is directly affected by the number of control loops to execute, which may have some limitations in terms of scalability. In order to address these issues, future work could consider alternative methods that can jointly optimise the different objectives. Such a problem was recently tackled from a theoretical perspective in [169], where a model based on game-theory was proposed to address the issue of interaction between load balancing and energy management applications. Future work could investigate how these principles could be applied in practice and develop a new approach that can provide more fairness between the different applications.

**Partitioning algorithm:** In order to partition the edge nodes into multiple sub-rings, the construction algorithm of the hybrid structure considers the geographical distribution of nodes in the network. However, other factors, which can be more specific to certain applications, may also be taken into account when forming the clusters. In the case of a cache management application, for instance, the number of sub-rings may influence the performance. In future extensions of this work, additional partitioning methods could be investigated and their impact on the performance of specific management applications could be evaluated.

**Monitoring infrastructure:** In order satisfy the requirements in terms of network awareness, the different adaptive resource management applications considered in this Ph.D. call for the support of monitoring mechanisms. Monitoring of dynamic conditions is a complicated subject that requires an independent infrastructure. The study of the impact of the monitoring system (e.g. monitoring accuracy or other capabilities) on the performance of the proposed approaches could be a subject of future work. In this thesis, it was assumed that the information management framework proposed in [83] could be used to support adaptive resource management. The framework relies on a distributed infrastructure, where a set of information management nodes, placed at different points in the network, can apply various functions, such as information collection, processing and dissemination. Future research could investigate how the proposed monitoring overlay could complement the management substrate infrastructure developed in this thesis. This would address issues that relate to determining the location and the number of information management nodes that are required in order to provide, in an effective manner, the relevant network information to the decision-making points. In addition, the use of different models of communication between the management substrate nodes and information management nodes, such as pull-based or push-based approaches, could be considered, and an evaluation of their impact on the accuracy of the information to be exchanged could be performed.

In order to address some of these issues, current research efforts, carried out in the context of the extension of this Ph.D., have been focusing on the implementation of the information management overlay ([83]) and the energy management application (chapter 5) in a testbed maintained in the Communications and Information Systems Group of the Electronic and Electrical Department of University College London.

# Bibliography


[1] Y. Diao, J. L. Hellerstein, S. Parekh, R. Griffith, G. E. Kaiser, and D. Phung. A control theory foundation for self-managing computing systems. *Selected Areas in Communications, IEEE Journal on*, 23(12):2213–2222, December.

[2] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. Overview and Principles of Internet Traffic Engineering. RFC 3272, May 2002.

[3] A. Atlas and A. Zinin. Basic Specification for IP Fast Reroute: Loop-Free Alternates. RFC 5286, September 2008.

[4] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer, IEEE*, 36:41–50, January 2003.

[5] M. Charalambides, G. Pavlou, P. Flegkas, Ning Wang, and D. Tuncer. Managing the future Internet through intelligent in-network substrates. *Network, IEEE*, 25(6):34–40, 2011.

[6] D. Tuncer, M. Charalambides, G. Pavlou, and N. Wang. Towards decentralized and adaptive network resource management. In *the proceedings of the 7th IEEE/IFIP International Conference on Network and Service Management (CNSM'11), mini-conference*, 2011.

[7] D. Tuncer, M. Charalambides, G. Pavlou, and N. Wang. DACoRM: A coordinated, decentralized and adaptive network resource management scheme. In *the proceedings of the IEEE Network Operations and Management Symposium (NOMS'12)*, pages 417–425, 2012.

[8] M. Charalambides, D. Tuncer, L. Mamatas, and G. Pavlou. Energy-aware adaptive network resource management. In *the proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM'13)*, pages 369–377, 2013.

[9] D. Tuncer, M. Charalambides, R. Landa, and G. Pavlou. More Control Over Network Resources: An ISP Caching Perspective. In *the proceedings of the 9th IEEE/IFIP International Conference on Network and Service Management (CNSM'13)*, 2013.

[10] J. Schonwalder, A. Pras, and J.P. Martin-Flatin. On the future of Internet management technologies. *Communications Magazine, IEEE*, 41(10):90–97, 2003.

[11] S. Van der Meer, W. Donnelly, J. Strassner, B. Jennings, and M.O. Foghlú. Emerging principles of autonomic network management. In *the proceedings of the 1st IEEE International Workshop on Modelling Autonomic Communications Environments (MACE'06)*, pages 29–48, 2006.

[12] R. Boutaba and J. Xiao. *Cognitive Networks: Towards Self-Aware Networks*, chapter Towards Self-Aware Networks, pages 77–95. Wiley, 2007.

[13] P. Horn. Autonomic computing: IBM's perspective on the state of information technology, 2001.

[14] The operators' vision on technologies, opportunities, risks and adoption roadmap, 2009. Study report, Autonomic Computing and Networking, Eurescom published study result, Study P1855, EDIN 0564-1855.

[15] A. Clemm. *Network Management Fundamentals*. Cisco Press, 2006.

[16] Cisco internetworking technology handbook - network management basics. `http://docwiki.cisco.com/wiki/Network_Management_Basics`. [Online; accessed 02-Dec-2013].

[17] D. Harrington, R. Presuhn, and B. Wijnen. An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks. RFC 3411, December 2002.

[18] G. Pavlou. On the evolution of management approaches, frameworks and protocols: A historical perspective. *Journal of Network and Systems Management*, 15(4):425–445, December 2007.

[19] M. Sloman. Policy driven management for distributed systems. *Journal of Network and Systems Management*, 2:333–360, 1994.

[20] P. Flegkas, P. Trimintzios, and G. Pavlou. A policy-based quality of service management system for IP DiffServ networks. *Network, IEEE*, 16(2):50–56, March 2002.

[21] N. Samaan and A. Karmouch. An automated policy-based management framework for differentiated communication systems. *IEEE Journal on Selected Areas in Communications*, 23(12):2236–2247, September 2006.

[22] N. Damianou, N. Dulay, E. Lupu, and M. Sloman. The ponder policy specification language. In *the proceedings of the International Workshop on Policies for Distributed Systems and Networks (POLICY'01)*, pages 18–38, 2001.

[23] D. Agrawal, KW. Lee, and J. Lobo. Policy-based management of networked computing systems. *Communications Magazine, IEEE*, 43(10):69–75, 2005.

[24] M. Charalambides, P. Flegkas, G. Pavlou, J. Rubio-Loyola, A. K. Bandara, E. C. Lupu, A. Russo, N. Dulay, and M. Sloman. Policy conflict analysis for DiffServ quality of service management. *IEEE Transactions on Network and Service Management*, 6(1):15–30, March 2009.

[25] M. Charalambides, P. Flegkas, G. Pavlou, J. Rubio-Loyola, A. K. Bandara, E.C. Lupu, A. Russo, M. Sloman, and N. Dulay. Dynamic policy analysis and conflict resolution for DiffServ quality of service management. In *the proceedings of the 10th IEEE/IFIP Network Operations and Management Symposium (NOMS'2006)*, pages 294–304, 2006.

[26] An Architectural Blueprint for Autonomic Computing. June 2001. `http://www-01.ibm.com/software/tivoli/autonomic/pdfs/AC_Blueprint_White_Paper_4th.pdf`. [Online; accessed 02-Dec-2013].

[27] J. O. Kephart. Research challenges of autonomic computing. In *the proceedings of the 27th International Conference on Software Engineering (ICSE'05)*, pages 15–22, 2005.

[28] M. C. Huebscher and J. A. McCann. A survey of autonomic computing - degrees, models, and applications. *ACM Computing Surveys*, 40(3):7:1–7:28, August 2008.

[29] M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.

[30] E. Lavinal, T. Desprats, and Y. Raynaud. A generic multi-agent conceptual framework towards self-management. In *the proceedings of the 10th IEEE/IFIP Network Operations and Management Symposium, (NOMS'06)*, pages 394 –403, April 2006.

[31] Z. Movahedi, M. Ayari, R. Langar, and G. Pujolle. A survey of autonomic network architectures and evaluation criteria. *Communications Surveys Tutorials, IEEE*, 14(2):464–490, 2012.

[32] S.R. White, J.E. Hanson, I. Whalley, D.M. Chess, and J.O. Kephart. An architectural approach to autonomic computing. In *the proceedings of the International Conference on Autonomic Computing (ICAC'04)*, pages 2–9, 2004.

[33] M. M. Fuad and M. J. Oudshoorn. System architecture of an autonomic element. In *the proceedings of the Fourth IEEE International Workshop on Engineering of Autonomic and Autonomous Systems*, pages 89–93, 2007.

[34] Y. Maurel, Ph. Lalanda, and A. Diaconescu. Towards introspectable, adaptable and extensible autonomic managers. In *the proceedings of the 7th IEEE/IFIP International Conference on Network and Service Management (CNSM'11), Short Paper*, 2011.

[35] R.M. Bahat, M.A. Bauer, E.M. Vieira, and O.K. Baek. Using policies to drive autonomic management. In *the proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks (WOWMOM'06)*, pages 475–479, 2006.

[36] R. M. Bahati, M. A. Bauer, and E. M. Vieira. Mapping policies into autonomic management actions. In *the proceedings of the International Conference on Autonomic and Autonomous Systems (ICAS'06)*, pages 38–, 2006.

[37] S. Davy, B. Jennings, and J. Strassner. Efficient policy conflict analysis for autonomic network management. In *the proceedings of the Fifth IEEE Workshop on Engineering of Autonomic and Autonomous Systems (EASE'08)*, pages 16–24, 2008.

[38] R. M. Bahati and M. A. Bauer. An adaptive reinforcement learning approach to policy-driven autonomic management. In *the proceedings of the 2009 Fifth International Conference on Autonomic and Autonomous Systems (ICAS'09)*, pages 135–141, 2009.

[39] H. Derbel, N. Agoulmine, and M. Salaün. ANEMA: Autonomic network management architecture to support self-configuration and self-optimization in IP networks. *Computer Networks*, 53:418–430, February 2009.

[40] J. O. Kephart and R. Das. Achieving self-management via utility functions. *IEEE Internet Computing*, 11(1):40–48, January 2007.

[41] G. Tesauro, R. Das, W. E. Walsh, and J. O. Kephart. Utility-function-driven resource allocation in autonomic systems. In *the proceedings of the Second International Conference on Autonomic Computing (ICAC'05)*, pages 342–343, 2005.

[42] N. Samaan. Achieving self-management in a distributed system of autonomic but social entities. In Sven Meer, Mark Burgess, and Spyros Denazis, editors, *Modelling Autonomic Communications Environments*, volume 5276 of *Lecture Notes in Computer Science*, pages 90–101. 2008.

[43] B. Jennings, S. van der Meer, S. Balasubramaniam, D. Botvich, M. O. Foghlu, W. Donnelly, and J. Strassner. Towards autonomic management of communications networks. *Communications Magazine, IEEE*, 45(10):112–121, October 2007.

[44] S. Dobson, S. Denazis, A. Fernández, D. Gaïti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, and F. Zambonelli. A survey of autonomic communications. *ACM Transactions on Autonomous and Adaptive Systems*, 1:223–259, December 2006.

[45] R. Mortier and E. Kiciman. Autonomic network management: some pragmatic considerations. In *the proceedings of the 2006 SIGCOMM workshop on Internet Network Management (INM'06)*, pages 89–93, 2006.

[46] N. Samaan and A. Karmouch. Towards autonomic network management: an analysis of current and future research directions. *Communications Surveys Tutorials, IEEE*, 11(3):22 –36, quarter 2009.

[47] Y. Cheng, R. Farha, M. S. Kim, A. Leon-Garcia, and J. Won-Ki Hong. A generic architecture for autonomic service and network management. *Computer Communications*, 29(18):3691–3709, November 2006.

[48] Univerself Project Website. `http://www.univerself-project.eu/`. [Online; accessed 02-Dec-2013].

[49] J. Strassner, N. Agoulmine, and E. Lehtihet. Focale: A novel autonomic networking architecture. In *the proceedings of the Latin American Autonomic Computing Symposium (LAACS'06)*, 2006.

[50] H. Liu and M. Parashar. Accord: a programming framework for autonomic applications. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 36(3):341–352, May.

[51] A. Tizghadam and A. Leon-Garcia. AORTA: Autonomic network control and management system. In *the proceedings of the IEEE INFOCOM Workshops (INFOCOM'08)*, pages 1–4, april 2008.

[52] The ANA blueprint, ANA Project - Deliverable D1.4/5/6v1 ANA Blueprint First Version, FP6-IST-27489. 2007. Technical report, ANA: Autonomic Network Architecture, A European Union funded project in Situated and Autonomic Communications.

[53] The Autonomic Internet (AutoI) Project Website. `http://ist-autoi.eu/`. [Online; accessed 02-Dec-2013].

[54] N. Koutsouris, K. Tsagkaris, P. Demestichas, Z. Altman, R. Combes, P. Peloso, L. Ciavaglia, L. Mamatas, S. Clayman, and A. Galis. Conflict free coordination of son functions in a unified management framework - demonstration of a proof of concept prototyping platform. In *the proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM'13), Demonstration Session Paper*, may 2013.

[55] D. D. Clark, C. Partridge, J. C. Ramming, and J. T. Wroclawski. A knowledge plane for the Internet. In *the proceedings of SIGCOMM 2003*, pages 3–10, 2003.

[56] A. Kind, X. Dimitropoulos, S. Denazis, and B. Claise. Advanced network monitoring brings life to the awareness plane. *Communications Magazine, IEEE*, 46(10):140–146, 2008.

[57] D. Gracanin, S.A. Bohner, and M. Hinchey. Towards a model-driven architecture for autonomic systems. In *the proceedings of the 11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems*, 2004.

[58] G. Tesauro, D. M. Chess, W. E. Walsh, R. Das, A. Segal, I. Whalley, J. O. Kephart, and S. R. White. A multi-agent systems approach to autonomic computing. In *the proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1 (AAMAS'04)*, pages 464–471, 2004.

[59] X. Dong, S. Hariri, L. Xue, H. Chen, M. Zhang, S. Pavuluri, and S. Rao. Autonomia: an autonomic computing environment. In *the proceedings of the 2003 IEEE International Performance, Computing, and Communications Conference*, April 2003.

[60] H. Chen, W. Zhou, and L. Liu. An approach of agent-based architecture for autonomic network management. In *the proceedings of the 5th International Conference on Wireless*

*Communications, Networking and Mobile Computing (WiCom'09)*, pages 1–5, September 2009.

[61] Y. Diao and A. Heching. Closed loop performance management for service delivery systems. In *the proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS'12)*, pages 61–69, 2012.

[62] J. Suzuki and T. Suda. A middleware platform for a biologically inspired network architecture supporting autonomous and adaptive applications. *IEEE Journal on Selected Areas in Communications*, 23(2):249–260, February.

[63] S. Balasubramaniam, D. Botvich, B. Jennings, S. Davy, W. Donnelly, and J. Strassner. Policy-constrained bio-inspired processes for autonomic route management. *Computer Networks*, 53(10):1666–1682, July 2009.

[64] O. Bonaventure, P. Trimintzios, G. Pavlou, B. Quoitin, A. Azcorra, M. Bagnulo, P. Flegkas, A. Garcia-Martinez, P. Georgatsos, L. Georgiadis, C. Jacquenet, L. Swinnen, and S. Tandel. *Quality of Future Internet Services, Cost263 final report*, chapter Internet Traffic Engineering, pages 118–179. Number 2856 in LNCS. Springer-Verlag, September 2003.

[65] N. Wang, KH. Ho, G. Pavlou, and M. Howarth. An overview of routing optimization for internet traffic engineering. *Communications Surveys Tutorials, IEEE*, 10(1):36 –56, quarter 2008.

[66] D. Mitra and K.G. Ramakrishnan. A case study of multiservice, multipriority traffic engineering design for data networks. In *the proceedings of the Global Telecommunications Conference, (GLOBECOM'99)*, volume 1B, pages 1077–1083, 1999.

[67] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *the proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'00)*, volume 2, pages 519 –528, 2000.

[68] B. Fortz, J. Rexford, and M. Thorup. Traffic engineering with traditional IP routing protocols. *Communications Magazine, IEEE*, 40(10):118 – 124, oct 2002.

[69] A. Sridharan, R. Guerin, and C. Diot. Achieving near-optimal traffic engineering solutions for current OSPF/IS-IS networks. *IEEE/ACM Transactions on Networking*, 13(2):234 – 247, April 2005.

[70] D. Xu, M. Chiang, and J. Rexford. Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering. *IEEE/ACM Transactions on Networking*, 19(6):1717–1730, 2011.

[71] S. Fischer, N. Kammenhuber, and A. Feldmann. REPLEX: dynamic traffic engineering based on wardrop routing policies. In *the proceedings of the ACM CoNEXT conference (CoNEXT'06)*, pages 1–12, 2006.

[72] A. Elwalid, C. Jin, S. Low, and I. Widjaja. MATE: MPLS adaptive traffic engineering. In *the proceedings of the 12th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'01)*, volume 3, pages 1300–1309, 2001.

[73] S. Kandula, D. Katabi, B. Davie, and A. Charny. Walking the tightrope: responsive yet stable traffic engineering. In *the proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM'05)*, volume 35, pages 253–264, August 2005.

[74] N. Wang, KH. Ho, and G. Pavlou. Adaptive multi-topology IGP based traffic engineering with near-optimal network performance. In Amitabha Das, Hung Pung, Francis Lee, and Lawrence Wong, editors, *NETWORKING 2008 Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*, volume 4982 of *Lecture Notes in Computer Science*, pages 654–666. Springer Berlin / Heidelberg, 2008.

[75] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright. Power awareness in network design and routing. In *the proceedings of the IEEE 27th Conference on Computer Communications (INFOCOM'08)*, pages 457–465, 2008.

[76] A. Coiro, F. Iervini, and M. Listanti. Distributed and adaptive interface switch off for internet energy saving. In *the proceedings of the 20th International Conference on Computer Communications and Networks (ICCCN)*, pages 1–8, 2011.

[77] F. Francois, N. Wang, K. Moessner, and S. Georgoulas. Optimizing link sleeping reconfigurations in isp networks with off-peak time failure protection. *IEEE Transactions on Network and Service Management*, pages 1–13, 2012.

[78] W. Jiang, R. Zhang-Shen, J. Rexford, and M. Chiang. Cooperative content distribution and traffic engineering in an ISP network. In *the proceedings of the 11th International Joint Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'09)*, pages 239–250, 2009.

[79] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, and K. K. Ramakrishnan. Optimal content placement for a large-scale vod system. In *the proceedings of the 6th ACM CoNEXT conference (CoNEXT'10)*, pages 1–12, 2010.

[80] V. Sourlas, P. Flegkas, L. Gkatzikis, and L. Tassiulas. Autonomic cache management in information-centric networks. In *the proceedings of the IEEE Network Operations and Management Symposium (NOMS'12)*, pages 121 –129, April 2012.

[81] F. Wuhib, M. Dam, R. Stadler, and A. Clem. Robust monitoring of network-wide aggregates through gossiping. *IEEE Transactions on Network and Service Management*, 6(2):95–109, 2009.

[82] F. Wuhib, R. Stadler, and M. Dam. Gossiping for threshold detection. In *the proceedings of the 11th IFIP/IEEE international conference on Symposium on Integrated Network Management (IM'09)*, pages 259–266, 2009.

[83] L. Mamatas, S. Clayman, M. Charalambides, A. Galis, and G. Pavlou. Towards an information management overlay for emerging networks. In *the proceedings of the 12th IEEE/IFIP Network Operations and Management Symposium (NOMS'10)*, pages 527–534, 2010.

[84] R. G. Clegg, S. Clayman, G. Pavlou, L. Mamatas, and A. Galis. On the selection of management/monitoring nodes in highly dynamic networks. *IEEE Transactions on Computers*, 62(6):1207–1220, 2013.

[85] N. Tcholtchev, M. Grajzer, and B. Vidalenc. Towards a unified architecture for resilience, survivability and autonomic fault-management for self-managing networks. In *the proceedings of the 2009 International Conference on Service-Oriented Computing (ICSOC/ServiceWave'09)*, pages 335–344, 2009.

[86] M. Jelasity, A. Montresor, and O. Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Transactions on Computer Systems*, 23(3):219–252, August 2005.

[87] F. Wuhib, R. Stadler, and M. Spreitzer. Gossip-based resource management for cloud environments. In *the proceedings of the 2010 International Conference on Network and Service Management (CNSM'10)*, pages 1–8, 2010.

[88] R. Makhloufi, G. Doyen, G. Bonnet, and D. Gaiti. Situated vs. global aggregation schemes for autonomous management systems. In *the proceedings of the 4th IFIP/IEEE*

*Workshop on Distributed Autonomous Network Management Systems*, pages 1135–1139, 2011.

[89] EK. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *Communications Surveys Tutorials, IEEE*, 7(2):72 – 93, quarter 2005.

[90] J. Risson and T. Moors. Survey of research towards robust peer-to-peer networks: search methods. *Computer Networks*, 50:3485–3521, December 2006.

[91] Z. Li and P. Mohapaira. The impact of topology on overlay routing service. In *the proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'04)*, volume 1, pages –418, 2004.

[92] M. Klein, B. Konig-Ries, and P. Obreiter. Service rings - a semantic overlay for service discovery in ad hoc networks. In *the proceedings of the 14th International Workshop on Database and Expert Systems Applications*, pages 180–185, 2003.

[93] S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon. Providing public intradomain traffic matrices to the research community. In *the proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM'06)*, volume 36, pages 83–86, January 2006.

[94] J. Postel. Transmission Control Protocol. RFC 793 (Standard), September 1981. Updated by RFCs 1122, 3168, 6093.

[95] S. Agarwal and J. R. Lorch. Matchmaking for online games and other latency-sensitive P2P systems. 39(4):315–326, August 2009.

[96] Y. Zhu, C. Dovrolis, and M. Ammar. Combining multihoming with overlay routing (or, how to be a better isp without owning a network). In *the proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM'07)*, pages 839 – 847, may 2007.

[97] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1982.

[98] Y. Crama, A.W.J. Kolen, and E.J. Pesch. Local search in combinatorial optimization. In P. J. Braspenning, F. Thuijsman, and A. J. M. M. Weijters, editors, *Artificial Neural*

*Networks*, volume 931 of *Lecture Notes in Computer Science*, pages 157–174. Springer Berlin Heidelberg, 1995.

[99] The Abilene topology and traffic matrices dataset, 2004. `http://www.cs.utexas.edu/~yzhang/research/AbileneTM/`. [Online; accessed 02-Dec-2013].

[100] The GEANT topology, 2004. `http://www.dante.net/server/show/nav.007009007`. [Online; accessed 02-Dec-2013].

[101] GNU Linear Programming Kit (GLPK). `http://www.gnu.org/software/glpk/`. [Online; accessed 02-Dec-2013].

[102] TOTEM Project: TOolbox for Traffic Engineering Methods. `http://totem.run.montefiore.ulg.ac.be/`. [Online; accessed 02-Dec-2013].

[103] S. Balon, J. Lepropre, O. Delcourt, F. Skivee, and G. Leduc. Traffic Engineering an Operational Network with the TOTEM Toolbox. *IEEE Transactions on Network and Service Management*, 4(1):51 –61, June 2007.

[104] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. Measuring ISP topologies with rocketfuel. *IEEE/ACM Transactions on Networking*, 12(1):2–16, February 2004.

[105] S. Knight, H.X. Nguyen, N. Falkner, R. Bowden, and M. Roughan. The Internet Topology Zoo. *IEEE Journal on Selected Areas in Communications*, 29(9):1765–1775, 2011.

[106] Abilene: the Internet2 Router Proxy Service. `http://routerproxy.grnoc.iu.edu/internet2/`. [Online; accessed 02-Dec-2013].

[107] The Backbone GEANT Looking Glass. `https://tools.geant.net/portal/links/lg/index.jsp`. [Online; accessed 02-Dec-2013].

[108] I. Gojmerac, P. Reichl, and L. Jansen. Towards low-complexity Internet traffic engineering: The Adaptive Multi-Path algorithm. *Computer Networks*, 52:2894–2907, October 2008.

[109] J.M. McQuillan, I. Richer, and E. Rosen. The New Routing Algorithm for the ARPANET. *IEEE Transactions on Communications*, 28(5):711–719, 1980.

[110] A. Khanna and J. Zinky. The revised ARPANET routing metric. In *the proceedings of the 1989 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM'89)*, volume 19, pages 45–56, August 1989.

[111] C. Villamazir. OSPF optimized multipath (OSPF-OMP). 1999. `http://tools.ietf.org/html/draft-ietf-ospf-omp-02`. [Online; accessed 02-Dec-2013].

[112] J. Moy. OSPF Version 2. RFC 2328 (Standard), April 1998. Updated by RFC 5709.

[113] A. Kvalbein, C. Dovrolis, and C. Muthu. Multipath load-adaptive routing: putting the emphasis on robustness and simplicity. In *the proceedings of the 17th IEEE International Conference on Network Protocols (ICNP'09)*, pages 203 –212, October 2009.

[114] P. Casas, F. Larroca, J.-L. Rougier, and S. Vaton. Robust routing vs dynamic load-balancing a comprehensive study and new directions. In *the proceedings of the 7th International Workshop on Design of Reliable Communication Networks (DRCN'09)*, pages 123 –130, October 2009.

[115] S. Sundaresan, C. Lumezanu, N. Feamster, and P. Francois. Autonomous traffic engineering with self-configuring topologies. In *the proceedings of the ACM SIGCOMM conference (SIGCOMM'10), Short Paper*, pages 417–418, 2010.

[116] M. Caesar, M. Casado, T. Koponen, J. Rexford, and S. Shenker. Dynamic route recomputation considered harmful. In *the proceedings of the 2010 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM'10)*, volume 40, pages 66–71, April 2010.

[117] P. Psenak, S. Mirtorabi, A. Roy, L. Nguyen, and P. Pillay-Esnault. Multi-Topology (MT) Routing in OSPF. RFC 4915 (Proposed Standard), June 2007.

[118] T. Przygienda, N. Shen, and N. Sheth. M-ISIS: Multi Topology (MT) Routing in Intermediate System to Intermediate Systems (IS-ISs). RFC 5120 (Proposed Standard), February 2008.

[119] S. Gjessing. Implementation of two resilience mechanisms using multi topology routing and stub routers. In *the proceedings of the Advanced International Conference on Telecommunications/International Conference on Internet and Web Applications and Services (AICT-ICIW'06)*, February 2006.

[120] M. Menth and R. Martin. Network resilience through multi-topology routing. In *the proceedings of the 5th International Workshop on Design of Reliable Communication Networks (DRCN'05)*, pages 271 – 277, October 2005.

[121] A. Kvalbein, A.F. Hansen, T. Cicic, S. Gjessing, and O. Lysne. Multiple routing configurations for fast IP network recovery. *IEEE/ACM Transactions on Networking*, 17(2):473 –486, April 2009.

[122] J. Wang, Y. Yang, L. Xiao, and K. Nahrstedt. Edge-based traffic engineering for OSPF networks. *Computer Networks*, 48:605–625, July 2005.

[123] A. Kvalbein and O. Lysne. How can multi-topology routing be used for intradomain traffic engineering? In *the proceedings of the SIGCOMM workshop on Internet Network Management (INM'07)*, pages 280–284, 2007.

[124] M. Laor and L. Gendel. The effect of packet reordering in a backbone link on application throughput. *Network, IEEE*, 16(5):28 – 36, September/October 2002.

[125] A. Asgari, R. Egan, P. Trimintzios, and G. Pavlou. Scalable monitoring support for resource management and service assurance. *Network, IEEE*, 18(6):6–18, November 2004.

[126] L. Liu, H. Wang, X. Liu, X. Jin, W. B. He, Q. B. Wang, and Y. Chen. GreenCloud: a new architecture for green data center. In *the proceedings of the 6th International Conference on Autonomic Computing and Communications- Industry Session (ICAC-INDST '09)*, pages 29–38, 2009.

[127] W. Fisher, M. Suchara, and J. Rexford. Greening backbone networks: reducing energy consumption by shutting off cables in bundled links. In *the proceedings of the first ACM SIGCOMM workshop on Green networking (Green Networking '10)*, pages 29–34, 2010.

[128] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall. Reducing network energy consumption via sleeping and rate-adaptation. In *the proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI'08)*, pages 323–336, 2008.

[129] F. Idzikowski, S. Orlowski, C. Raack, H. Woesner, and A. Wolisz. Saving energy in IP-over-WDM networks by switching off line cards in low-demand scenarios. In *the proceedings of the 14th conference on Optical Network Design and Modeling (ONDM'10)*, pages 42–47, 2010.

[130] F. Cuomo, A. Cianfrani, M. Polverini, and D. Mangione. Network pruning for energy saving in the internet. *Computer Networks*, 56(10):2355–2367, July 2012.

[131] F. Giroire, D. Mazauric, J. Moulierac, and B. Onfroy. Minimizing routing energy consumption: From theoretical to practical results. In *Green Computing and Communications (GreenCom), 2010 IEEE/ACM International Conference on Cyber, Physical and Social Computing (CPSCom)*, pages 252–259, 2010.

[132] M. Zhang, C. Yi, B. Liu, and B. Zhang. Greente: Power-aware traffic engineering. In *the proceedings of the 18th IEEE International Conference on Network Protocols (ICNP'10)*, pages 21–30, 2010.

[133] G. Athanasiou, K. Tsagkaris, P. Vlacheas, and P. Demestichas. Introducing energy-awareness in traffic engineering for future networks. In *the proceedings of the 7th International Conference on Network and Services Management (CNSM'11)*, pages 367–370, 2011.

[134] A. Cianfrani, V. Eramo, M. Listanti, M. Polverini, and A.V. Vasilakos. An OSPF-integrated routing strategy for QoS-aware energy saving in IP backbone networks. *IEEE Transactions on Network and Service Management*, 9(3):254–267, 2012.

[135] H. Matsuura. Energy-saving routing algorithm using Steiner tree. In *the proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM'13)*, May 2013.

[136] A. P. Bianzino, L. Chiaraviglio, M. Mellia, and JL. Rougier. GRiDA: GReen Distributed Algorithm for energy-efficient IP backbone networks. *Computer Networks*, 56(14):3219 – 3232, 2012.

[137] S. S. W. Lee, Po-Kai Tseng, and A. Chen. Multi-topology design and link weight assignment for green IP networks. In *the proceedings of the 2011 IEEE Symposium on Computers and Communications (ISCC'11)*, pages 377–382, 2011.

[138] IEEE Standard 802.1AX: Link Aggregation, 2008.

[139] R. Bolla, R. Bruschi, A. Cianfrani, and M. Listanti. Enabling backbone networks to sleep. *Network, IEEE*, 25(2):26–31, March 2011.

[140] AJ. Su, David R. Choffnes, A. Kuzmanovic, and F. E. Bustamante. Drafting behind Akamai (travelocity-based detouring). In *the proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM'06)*, volume 36, pages 435–446, August 2006.

[141] Akamai CDN. `http://www.akamai.com`. [Online; accessed 02-Dec-2013].

[142] Limelight Networks CDN. `http://www.limelight.com`. [Online; accessed 02-Dec-2013].

[143] B. Frank, I. Poese, G. Smaragdakis, S. Uhlig, and A. Feldmann. Content-aware traffic engineering. In *the proceedings of the 12th ACM Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'12), Short Paper*, pages 413–414, 2012.

[144] Content Delivery Networks Interconnection (CDNI). `http://tools.ietf.org/wg/cdni/draft-ietf-cdni-framework/`. [Online; accessed 02-Dec-2013].

[145] Coral CDN. `http://www.coralcdn.org`. [Online; accessed 02-Dec-2013].

[146] P. Wendell, J. W. Jiang, M. J. Freedman, and J. Rexford. DONAR: decentralized server selection for cloud services. In *the proceedings of the 2010 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM'10)*, volume 40, pages 231–242, August 2010.

[147] N. Kamiyama, T. Mori, R. Kawahara, S. Harada, and H. Hasegawa. ISP-Operated CDN. In *the proceedings of IEEE INFOCOM Workshops 2009*, pages 1–6, April 2009.

[148] K. Cho, H. Jung, M. Lee, D. Ko, T. T. Kwon, and Y. Choi. How can an ISP merge with a CDN? *Communications Magazine, IEEE*, 49(10):156–162, October 2011.

[149] L. Qiu, V. N. Padmanabhan, and G. M. Voelker. On the placement of web server replicas. In *the proceedings of the 12th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'01)*, volume 3, pages 1587–1596, 2001.

[150] J. Kangasharju, J. Roberts, and K. W. Ross. Object replication strategies in content distribution networks. *Computer Communications*, 25(4):376–383, March 2002.

[151] N. Laoutaris, O. Telelis, V. Zissimopoulos, and I. Stavrakakis. Distributed Selfish Replication. *IEEE Transactions on Parallel and Distributed Systems*, 17(12):1401–1413, December 2006.

[152] J. Dai, Z. Hu, B. Li, J. Liu, and B. Li. Collaborative hierarchical caching with dynamic request routing for massive content distribution. In *the proceedings of Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'12)*, pages 2444 – 2452, March 2012.

[153] M.R. Korupolu and M. Dahlin. Coordinated placement and replacement for large-scale distributed caches. *IEEE Transactions on Knowledge and Data Engineering*, 14(6):1317–1329, Nov./Dec. 2002.

[154] S. Borst, V. Gupta, and A. Walid. Distributed caching algorithms for content distribution networks. In *the proceedings of Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'10)*, pages 1–9, March 2010.

[155] BG. Chun, K. Chaudhuri, H. Wee, M. Barreno, C. H. Papadimitriou, and J. Kubiatowicz. Selfish caching in distributed systems: a game-theoretic analysis. In *the proceedings of the 23rd annual ACM Symposium on Principles of Distributed Computing (PODC'04)*, pages 21–30, 2004.

[156] I. Baev, R. Rajaraman, and C. Swamy. Approximation algorithms for data placement problems. *SIAM Journal on Computing*, 38(4):1411–1429, 2008.

[157] T. Bektaş, JF. Cordeau, E. Erkut, and G. Laporte. Exact algorithms for the joint object placement and request routing problem in content distribution networks. *Computers & Operations Research*, 35(12):3860–3884, December 2008.

[158] BBC iPlayer. `http://www.bbc.co.uk/iplayer/`. [Online; accessed 02-Dec-2013].

[159] S. Balon, F. Skivée, and G. Leduc. How well do traffic engineering objective functions meet te requirements? In *the proceedings of the IFIP-TC6 NETWORKING'06*, pages 75–86, 2006.

[160] A. Pathan and R. Buyya. A taxonomy and survey of content delivery networks, February 2007. Technical Report, GRIDS-TR-2007-4, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia, `http://www.gridbus.org/reports/CDN-Taxonomy.pdf`. [Online; accessed 02-Dec-2013].

[161] H. Yu, D. Zheng, B.Y. Zhao, and W. Zheng. Understanding user behavior in large-scale video-on-demand systems. In *the proceedings of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems (EuroSys'06)*, pages 333–344, 2006.

[162] G. Rossini D. Rossi. Caching performance of content centric networks under multi-path routing (and more). Telecom ParisTech, Paris, France.

[163] N. Tcholtchev, R. Chaparadza, and A. Prakash. Addressing stability of control-loops in the context of the GANA architecture: Synchronization of actions and policies. In *the proceedings of the 4th IFIP TC 6 International Workshop on Self-Organizing Systems (IWSOS'09)*, pages 262–268, 2009.

[164] M. Litoiu, M. Woodside, and T. Zheng. Hierarchical model-based autonomic control of software systems. In *the proceedings of the 2005 Workshop on Design and Evolution of Autonomic application Software (DEAS'05)*, pages 1–7, 2005.

[165] SW. Cheng, AC. Huang, D. Garlan, B. Schmerl, and P. Steenkiste. An architecture for coordinating multiple self-management systems. In *the proceedings of the Fourth Working IEEE/IFIP Conference on Software Architecture (WICSA'04)*, pages 12–15, 2004.

[166] Y. Brun, G. Marzo Serugendo, C. Gacek, H. Giese, H. Kienle, M. Litoiu, H. Müller, M. Pezzè, and M. Shaw. Software Engineering for Self-Adaptive Systems. chapter Engineering Self-Adaptive Systems through Feedback Loops, pages 48–70. Springer-Verlag, Berlin, Heidelberg, 2009.

[167] R. T. Marler and J. S. Arora. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26(6):369–395, 2004.

[168] A. Konstantinidis, D. Zeinalipour-Yazti, P. Andreou, and G. Samaras. Multi-objective query optimization in smartphone social networks. In *the proceedings of the 12th IEEE International Conference on Mobile Data Management (MDM'11)*, volume 1, pages 27–32, 2011.

[169] Y. Zhao, S. Wang, S. Xu, X. Wang, X. Gao, and C. Qiao. Load Balance vs Energy Efficiency in Traffic Engineering: A Game Theoretical Perspective. In *the proceedings of the 32nd IEEE International Conference on Computer Communications (INFOCOM'13), mini-conference*, 2013.